Juho Jokela

Evaluating and measuring the adequacy of a programming job applicant

Using code tests as a method of evaluation

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Media Technology

Opinnäytetyö/Thesis

20.4.2017

Metropolia

Author	Juho Jokela
Title	Evaluating and measuring the adequacy of a programming job applicant
Number of Pages	61 pages + 5 appendices
Date	Thursday 20th April, 2017
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital media
Instructors	Niklas Huotari, CEO Patrick Ausderau, Lecturer

Code tests are a widely used method in the information technology (IT) recruitment process, as they ease the technical assessment of candidates. The goal of this thesis is to identify how their questions should be structured. The functionality of code tests is also examined both in individual job openings and as a part of the preliminary applicant grading of Coding Factory 2017 recruitment event. These information are valuable as the client company could use them in further development of code tests for similar events. Achieving a comprehensive view on code tests shows their possible potential and when best to utilise them.

The thesis was conducted in the office of the client company and the main software used were Microsoft Word and Excel applications. These were used to segment the questions into certain categories and reveal the usability of code tests as a prequalification method on the event applicants.

The existing code test questions found from platforms were compared to one another using a standard testing method (CRESST). From these results the correct structure of questions was mapped out. Other material utilised were the aforesaid event and its predecessor's documentation and the results of an event participant interview. These were used in an analysis to weight the pros and cons of code tests.

The end product of this research is this study. It guides the test developers to build more functional code test questions. This research also advices when to use code tests as a part of the recruitment process. Based on this thesis, there are a lot to uncover in code tests still, especially from the view of the applicants.

Keywords	information technology, job recruitment, code tests

Tekijä Otsikko Sivumäärä Aika	Juho Jokela Ohjelmoivan työnhakijan arviointi ohjelmointitestien avulla 61 sivua + 5 liitettä 20.4.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaajat	Toimitusjohtaja Niklas Huotari Lehtori Patrick Ausderau

Ohjelmointitestejä hyödynnetään nykyään laajalti osana rekrytointiprosessia, koska niiden avulla pystytään arvioimaan hakijan konkreettista ohjelmointitaitoa tarkemmin kuin millään muulla menetelmällä. Insinöörityön tarkoitus oli selvittää, kuinka ohjelmointitestien kysymykset tulisi rakentaa. Lisäksi työssä tutkittiin ohjelmointitestien toimivuutta arviointimenetelmänä, niin yksittäisissä hauissa kuin osana Coding Factory 2017 -rekrytointitapahtumaan hakeneiden arviointiprosessia. Työssä pyrittiin selkeyttämään kysymysten oikeanlainen rakenne, jotta saatua tietoa voitaisiin hyödyntää sekä asiakasyrityksen tulevaisuuden tapahtumissa että rekrytoijien apuvälineenä teknisten haastattelujen kehittämisessä. Ohjelmointitestien yleinen arviointi havainnollistaa niiden mahdollisen potentiaalin ja oikean käyttöyhteyden.

Insinöörityö toteutettiin asiakasyrityksen tiloissa, ja pääasialliset työkalut olivat Microsoftin Word- ja Excel-sovellukset. Näillä sovelluksilla jaoteltiin kysymykset kategorioihinsa ja arvioitiin ohjelmointitestien hyöty hakijoiden karsintamenetelmänä tapahtumissa.

Työssä vertailtiin olemassa olevien alustojen tarjoamien ohjelmointitestien kysymyksiä toisiinsa hyödyntäen opettajien käyttämää CRESST-menetelmää. Sen tulosten pohjalta kartoitettiin, kuinka toimivat kysymykset tulisi rakentaa ja mitä asioita on huomioitava. Muita hyödynnettyjä materiaaleja olivat edellä mainitun tapahtuman ja sen edeltäjän dokumentaatiot sekä tapahtumaan osallistuneen hakijan haastattelu, joista laadittiin SWOT-analyysi yleisten hyötyjen arvioimiseen. Tämän menetelmän tulosten perusteella voidaan todeta, että ohjelmointitestit ovat muun muassa arvokas lisätyökalu rekrytoijille, mutta työläitä niiden suunnittelijoille ja kehittäjille.

Tutkimuksen lopputuote on tämä opinnäytetyö. Sen avulla ohjelmointitesteistä pystytään luomaan laadukkaampia. Tutkimus myös ohjeistaa rekrytoivia yrityksiä siitä, milloin ohjelmointitestejä tulisi käyttää osana rekrytointiprosessia. Ohjelmointitestien hyöty kasvaa suhteessa hakijoiden määrään: isojen henkilömäärien arviointi ilman automatisoituja menetelmiä on työlästä, ja tähän ongelmaan ohjelmointitestit tarjoavat ratkaisun.

Avainsanat	informaatiotekniikka, rekrytointi, ohjelmointitestit

Contents

Abbreviations

B-APT	the Berger Aptitude for Programming Test.
B-APT AF	the Berger Aptitude for Programming Test Advanced Form.
CRESST	National Center for Research on Evaluation, Standards, and Student Testing.
EB	Employer Branding.
HR	Human Resources.
HTML	HyperText Markup Language.
ICT	information and communications technology.
IoT	the Internet of Things.
IT	information technology.
SNS	Social Networking Sites.

Glossary

code test	a test consisting of programming related questions that measure candidates programming capabilities.
E-recruitment	a form of recruiting where Internet is used in some form to assist conventional recruitment .
headhunting	Identifying, locating and eventually convincing an employee to leave his/her current employer to join a competing organization.
meter	the mathematical definition of a metric is "a rule used to describe how far apart two point are. More formally, a metric is a function m defined on pairs of objects x and y such that m(x,y) represents the distance between x and y" [1].
skill	a skill is **a learned ability** to carry out a task with pre-determined results often within a given amount of time, energy, or both. In other words the abilities that one possesses. Skills can often be divided into domain-general and domain-specific skills [2, p. 4].

1 Introduction

The goal of my thesis is to broadly examine IT recruitment and its processes and to find means to improve one of the most commonly used part of the process, code tests. Achieving this would result in a baseline scheme, useful in detecting when the use of a code test should be advised and what types of questions it should consist of. The research aims to find out the advantages and drawbacks of using a code test, focus in differentiating functional questions from poorly made ones and aim to reveal some rules that could be used to construct an efficient code test for an event, as it has a larger quantity of candidates in comparison to a regular job opening, therefore making certain assessment tools ineffective.

The research questions are selected based on their relevance to educating the recruiters of aTalent Recruiting, the client company of this research, and to advise in the further development of the code tests used in the evaluation model of Coding Factory 2017, an event organized by aTalent Recruiting that was held in January 28, 2017 in Teatteri Toivo, Helsinki. The code tests utilised in this event were structured specifically for it by a team of developers, of which I was a part of. The reason why familiarizing and educating recruiters on information technology is essential stems from its current significance in the recruitment industry.

Modern society is digital and there is an endeavour to digitalizing everything that can be digitalized. There were almost 22 billion devices connected to the Internet in 2016 and Statista, one of the leading statistics companies on the Internet, approximates that in 2019 that number could rise to 50 billion [3]. Nowadays programming is required to produce content for all sorts of devices and good programmers are valued highly. There is an ever growing demand for new software and therefore, IT companies are trying to master recruitment in order to have the workforce meet the demand.

Nowadays both parties, the employee and employer, value each other. The companies are looking for indispensable employees that understand the organization's mission and vision and meet both the personal and organizational goals. The employees are looking for more than just the salary in a job, such as flexibility and variance in the job description or company benefits. [4, pp. 20-24, p. 34]

A concrete example of mutual respect is the trend in which IT consulting companies are currently hiring. They seek for and hire young talents and then gradually make their job descriptions more demanding as their skills improve. In this case, the companies promise tutoring, interesting job opportunities and an overall good workplace, and in return they expect their employees to stay in the company for years while developing both themselves and the company. Nonetheless, the challenge is to find and evaluate the applicants effectively.

2 Theoretical background

The theoretical part of this thesis aims to familiarize the reader with recruitment and the tools and practises used when assessing candidates for job openings. The sections are to introduce the ways recruitment can be done, its contents, the industry, the available methods and their utilization and structuring for organization's needs, the origins of candidate evaluation models and the thought process that goes into candidates' skill sourcing and capabilities. Because code tests are in the focus of interest in this thesis, they are being examined more attentively. Lastly, the model that was used in Coding Factory 2017, the code tests of which this thesis aims to offer improvements on for future events, is presented briefly.

2.1 Overview of the state of programming

In the year 2014 there were roughly 29 million information and communications technology (ICT)-skilled workers and 18 million software developers [5]. The numbers may seem sufficient but are not, due to the quick advancement of programming languages and technologies and the upcoming labour shortage caused by the elderly workforce retiring in the near future. According to the employers, the majority of developers seeking work are not at all competent and many programmers simply lack the skills and the motivation to keep developing themselves and keep in par with the evolution of technology [6, p. 10]. Of course, the trend of saying that the programmers currently in the job market are simply "inadequate and uneducated" is not at all new and has been around for decades [7]. Technology is ever evolving and it is only inevitable that at one point the programmers of today will no longer meet the requirements of tomorrow.

The fact that companies are reluctant to lower their standards is a problem. In order for an organization to grow, the workforce also has to increase. Unwillingness to invest capital in educating the new employees to the company's technologies will result in losing the most promising candidates to companies that do offer these services. The more a company is willing to invest in their employees, the more they are ready to invest in the company, resulting in directly increasing the value of the company.

As an example of IT’s evolution, the coming of cloud computing and the Internet of Things (IoT) have revolutionized virtual data storage and gathering, and new programming languages and frameworks are being introduced, providing innovative solutions. The direct effect of the constant advancement of technology is that the software built on older platforms needs to be updated and modernized as well, generating work for programmers.

However, there are some challenges in the recruiting of programmers that are caused both by the employers and the job applicants. The next section presents the methods available for human resourcing and recruitment and elucidates the process, helping in understanding the origin of these issues.

2.2 Recruitment

Recruitment can be described as “the process of attracting, screening, and selecting a qualified person (from within or outside of an organization) for a job opening” [8, p. 1] and in practise it consists of publishing job advertisements, reviewing job applications, interviewing and potentially testing the applicants, and eventually leading to hiring or presenting the candidates to the client company.

E-recruitment is a form of recruitment where Internet is used in some form to assist conventional recruitment. It enables organizations to shorten their hiring times through increased information flow and accelerated recruitment processes, making recruitment in general more affordable and efficient. E-recruitment also significantly eases global hiring, extending the reach of a job advertisement. When referring to recruitment, e-recruitment is perceived as a part of it and not as a separate component. As code tests are often online, they can therefore be considered a part of e-recruitment. [9, pp. 4-5]

Recruitment is a potently growing and competitive line of business. The recruitment industry has been steadily growing, and in 2015 the global annual growth was 8.6% with sales revenues of €450 billion [10]. One of the biggest sectors of recruitment that has seen a lot of growth recently is that of engineers [11]. Unfortunately, engineers are also one of the hardest to hire [12, p. 7] [13]. One reason for that is that there is a shortage of competent programmers and for companies it is very difficult to distinguish themselves from the competitors and appear more appealing. Exaggerating a little by generalizing, the situation is such that advanced programmers can almost choose where they wish to work. Hiring people in some branches of science has become more difficult, as employees have higher workplace standards, which leads to companies enhancing and improving their Employer Branding (EB) and social media advertising. To compensate for the competition, recruiters and the whole Human Resources (HR)-sector aim to be more active and try to make the company recognizable, stand-out and well known.

Before making the decision to hire new employees, it is critical to evaluate how beneficial the new hire would be for the company. Lack of knowledge regarding the current employees' strengths and team structure will harm the company, leading to either a waste of time, when, after several candidate meetings and interviews, the decision to not hire anyone is eventually made, or to an inadequate hire, as the company decides to hire someone whose talents are not the ones needed. Defining the job-description and the needs of the company as specifically as possible and then trimming the requirements to just the bare minimum is easy to execute and a solid strategy when finding out what would make the company's workforce grow. The strength of this strategy is that the important parts of what the job consists of get highlighted, which helps recruiters identify what is necessary and what are the more irrelevant details that can be used to make distinctions between candidates.

Bad or failed hires lead to loss of money, deteriorate the office's atmosphere and harm the brand. A successful hiring brings new energy and workforce to the office and increases both efficiency and quality of work. New employees are expected to be fast learners and adaptive to a variety of tasks [14, p. 100]. They may also open new channels for HR with their connections to universities or previous workplace co-workers, to name a few. It is a good practise to map out the new employee's social network and try to figure out if the employee's contacts could potentially be of help when recruiting again. [15]

2.2.1 Recruitment process

The whole recruitment process depends a lot on the company and the job-opening and there is a lot of margin for variance. There is not a single mold that would fit to every company or role there is, yet process length can be utilised as a meter for segregating the recruitment processes into smaller contexts.

In some cases, what a company needs is any kind of an employee as fast as possible. A role like this could be, for example, the position of a cashier, or any other kind of job that can be done by someone with a low level of education. In these cases, it is not worth the effort, time and money to go through a long process, but instead saving resources by getting a fast hire is all that matters. When a company wants someone for a specific position that is either intellectually or otherwise demanding, the process will get lengthier. As the requirements grow, so does the length of the recruitment process.

In the best case, a cashier's position can be filled within days but to find someone in the same time period for any role that requires a highly educated employee of any profession is practically impossible without an existing database that consists of potential candidates. This is because as the requirements of the job description increase, the amount of people able to fill those requirements inversely decreases in proportion.

The process of constructing a job advertisement for an IT position has some unique features due to the intense competition between the companies. What really stands out from a lot of other lines of work is that the mandatory job requirements are usually forced to go down a lot in order to achieve a broad pool of candidates to choose from [6, p. 48]. There are often ten different companies looking for exactly the same type of candidate and being swift and flexible with the hiring decisions and making the evaluation pipe as candidate-friendly as possible is a high priority. Initially a company may be looking for a candidate with three or more years of experience, yet, in comparison to some other candidate with only one year of experience but a lot of other expertise or spare time activity, the difference is not so substantial that the candidate should be ignored.

When creating the advertisement, screening of the alternative technologies available, ones that are relatively similar to what the company is using, is advised. The reasoning for this is that knowledge of one programming language can be adapted to another. Also, the team size and structure play a big role when building advertisements and teams that have the ability to guide and educate the new workforce are often seen as more approachable and appealing to candidates. [16]

The main feature of a well planned and executed recruitment process is that it is as streamlined as possible. Both the company recruiting and the applicants should be kept in mind and in an ideal case each party would be satisfied after the process. The biggest downfall of a recruitment process is making the evaluation model overly complex. Use of committees, approval steps and staying distant will end up inconveniencing all parties. Having a lengthy and overly complex recruitment process will lead to candidates dropping out and to other companies snatching the most prominent candidates. [6, p. 42]

In figure 1 there is a simplistic and efficient evaluation pipe. For a lot of positions, the following evaluation pipe is insufficient. However, it is good to keep it in mind and then, block by block, build more phases depending on the difficulty of the job description.

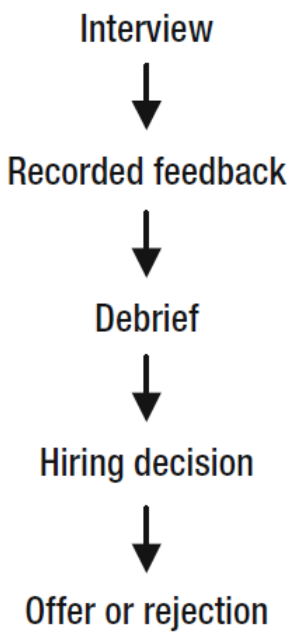


Figure 1: A short, straightforward evaluation pipe. Copied from McCuller (2012) [6, p. 42].

As mentioned, having accurate and thorough understanding of the company and its team structure is important. The process pipeline should be built based on this knowledge. In figure 2, there is an example of a high-level candidate pipeline from the point of view of the hiring company.

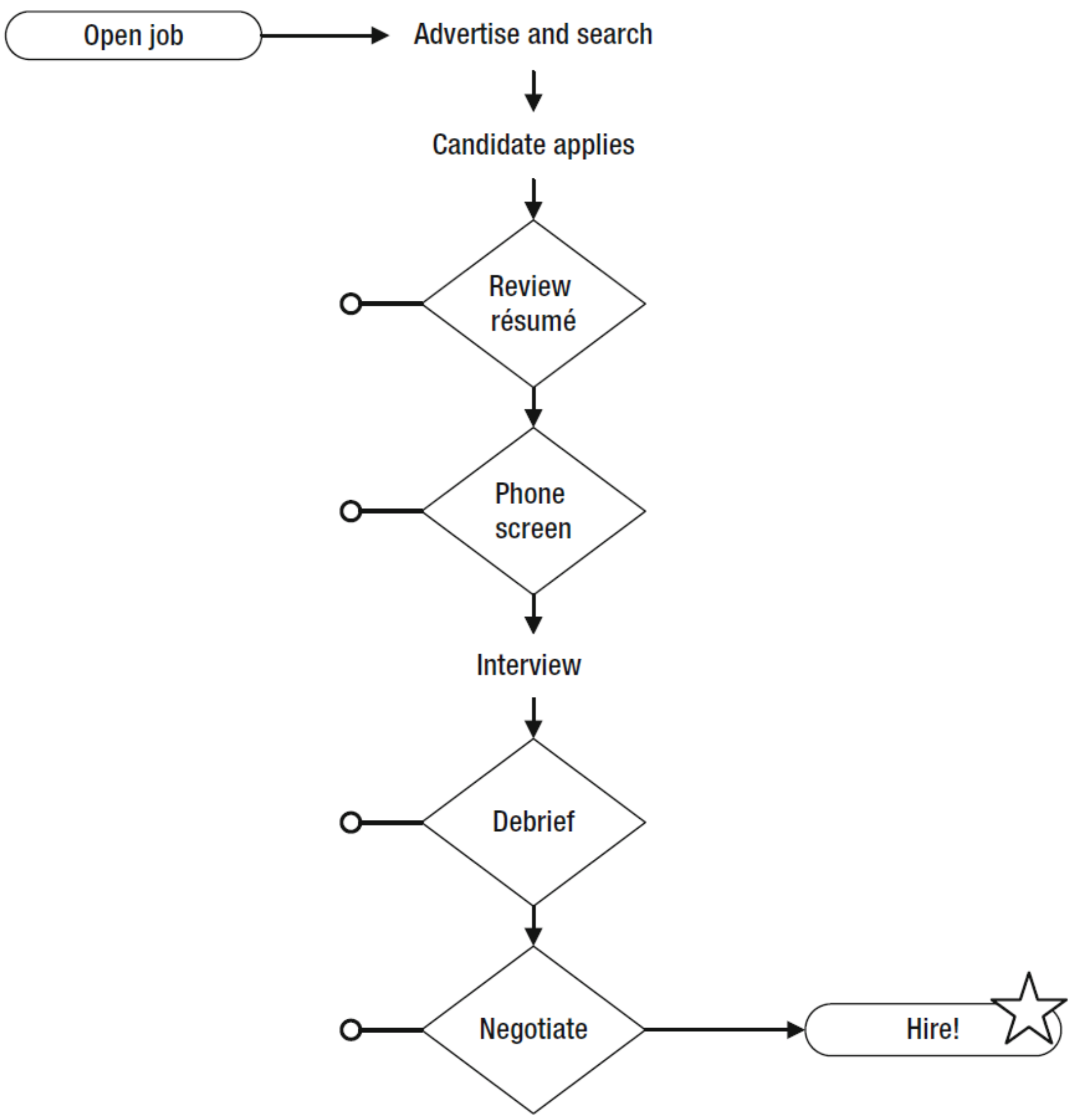


Figure 2: An example of a high-level candidate pipeline. Copied from McCuller (2012) [6, p. 28].

In IT job openings, especially programming, it is sometimes beneficial to turn the evaluation process upside down, for example by starting with a code test and then interviewing the candidates that did well in the test. A simple code test can eliminate a lot of candidates, while being relatively easy to evaluate and taking a lot less time than going through each candidates' résumés and profiles such as the ones that the candidate may have in programming websites like GitHub and Stack Overflow. Another option is to leave the application letter out to establish an easier applying process and to make it more candidate-friendly. The lower the bar on applying, the more applications there are in total. [16]

What candidates appreciate the most is that the job advertisement is clear and the evaluation process is as transparent as possible, so that the candidates are aware of their situation. Receiving feedback is also valued highly. [17]

A candidate always has the least knowledge about the company's pipeline and therefore informing them about what steps there are is a way to prevent misunderstandings [6, p. 26]. An example of what a typical pipeline from a candidate's perspective looks like is displayed in figure 3.

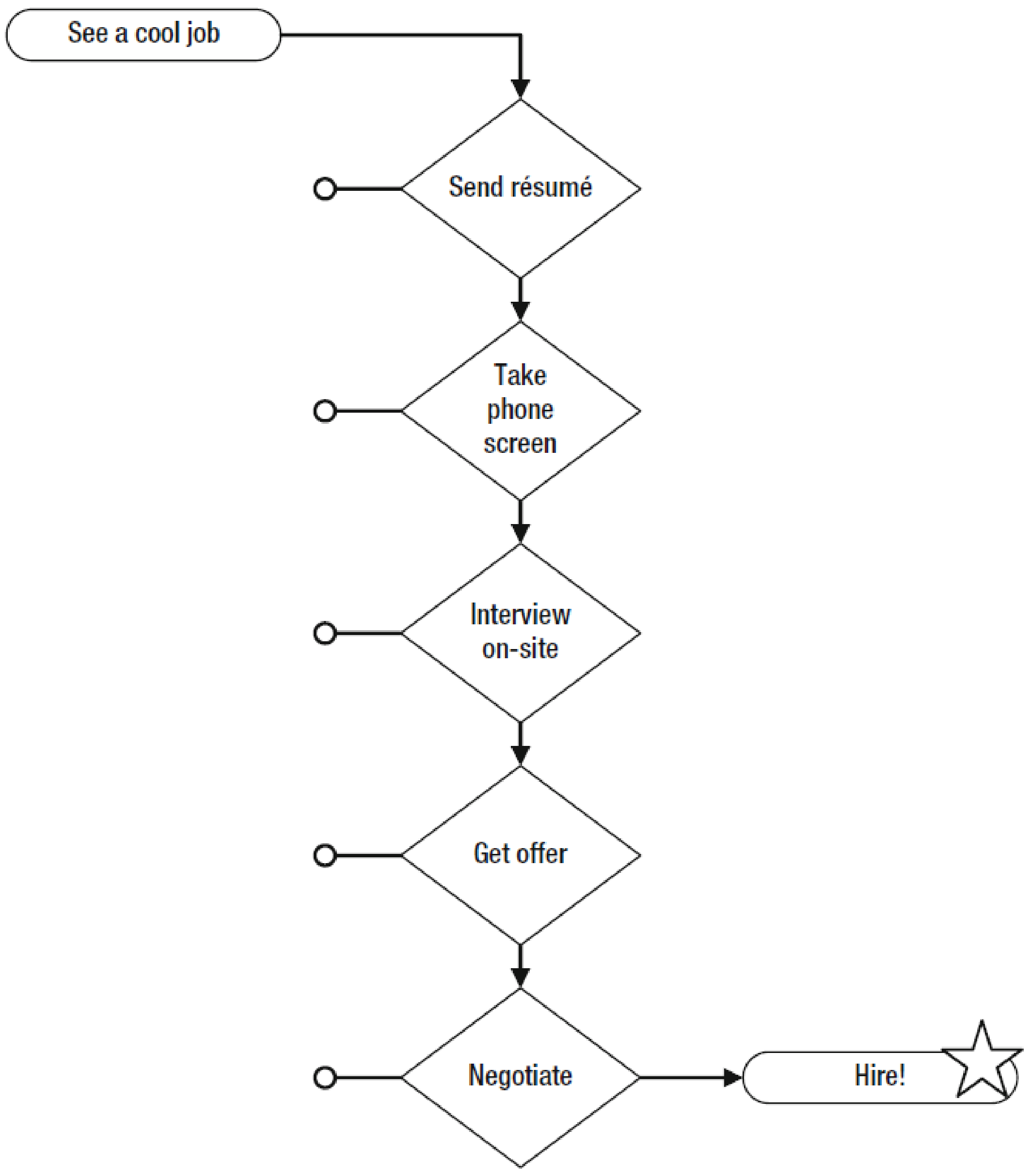


Figure 3: A candidate's perspective on their lifecycle in a typical pipeline. Copied from McCuller (2012) [6, p. 27].

The phases of which the pipe in both figures, 2 and 3, consist of are some of the most common. The résumés are for overall candidate evaluation, often followed by a phone screen, that is, a phone interview, and an actual face-to-face interview, to map the candidates social skills and behaviour, and then a negotiation on the job terms, such as salary and work hours. When comparing the candidate's perspective in figure 3, to the company's pipeline in figure 2, there are no major differences and both parties clearly know what the process consists of and what is the order in which the process progresses. This is an example of an ideal case where informing the candidate has gone well.

After the applying deadline has passed, if there is any, and the basic screening has been done and all candidates' portfolios, CVs and application letters have been reviewed and the recruiter has come to a conclusion as to which candidates should be proceeding in the recruitment process, it is good to inform all candidates whether they have been approved to move onwards in the recruitment process or declined. It is important for EB that all applicants are treated as customers as it results in making the general view of the company more positive and the declined candidates will more likely apply for other positions that may open in the future. [6, p. 32]

2.2.2 Finding of suitable candidates

While there are multiple different kinds of candidates, it is possible to define candidacy as the following: a candidate is someone who "may be actively looking for a job, passively looking, not looking but open to moving, not looking and actually quite content, or looking for a future position (such as after graduation)" [6, p. 24].

When searching for and contacting candidates, activity is the key. When a potential candidate turns up, the recruiter will immediately try contacting them directly so as to be ahead of competitors. Most of the times recruiters are not going to get a direct response to their emails and a response rate of 20% can be considered very good. When recruiters are contacting suitable candidates, they are, according to Ardy Daie, "planting seeds", that is, must keep in mind that "not everyone is going to respond to (them) right away". This means that the companies that contact candidates will be remembered and that persistence will eventually lead to results. [15]

Using a combination of channels such as headhunting and job advertisements is the optimal way of reaching as many potential candidates as possible. The more channels available, usually the better the candidates found, but if the job opening or the company is not appealing, it is best to strip the channels to a certain level to save resources.

Posting the job advertisement to a company's website is the cheapest available channel, while headhunting is generally the most expensive option [18, p. 21]. If the company is not well known and the amount of visitors in the company's website is low then the use of recruitment companies is also a valid option. The main benefits of using recruitment companies is that they have existing databases containing candidates as well as widespread connections to job boards, unions and subject associations, to name a few. The price in which recruitment companies offer their services varies but for comparison just a single job advert to a job board can be hundreds of euros [19]. In the case of the Coding Factory 2017 event, the participating organizations outsourced the advertising, candidate sourcing and the initial elimination process to a recruitment company to ensure that there were enough valid participants, while not requiring a lot of resources spent from the companies themselves.

Another affordable route for finding and interacting with candidates is via Social Networking Sites (SNS). Initially intended for socialising with relatives and friends, these networks are an important tool for recruitment professionals. SNSs can be used for marketing, both for employers and job applicants, and as a screening mechanism, providing a lot of information about the job seeker [9, p. 5].

Websites LinkedIn, GitHub, BitBucket and Stack Exchange are the main platforms that candidates use to publish information about themselves. These websites can be considered the single most reliable source for making deductions consisting of recommendations, previous workplaces and most importantly concrete proof of IT knowledge [6, pp. 6-7].

A candidate may have a profile in each of these websites, and the recruiters can view these to obtain information about a candidate. In LinkedIn, a recruiter may find the candidate's basic information, such as their name and occupation, work experience, education and other sections that would be in a traditional résumé or CV [20].

Websites GitHub and BitBucket utilize Git, an open source version control system, and programmers use these websites for storing their projects [21]. Unlike LinkedIn, the information stored on these website is mostly pure code, meaning it can be difficult to make deductions or to make sense of the candidate's profile. Knowing how to utilize these websites, however, can tell a lot about a programming candidate.

Stack Exchange is a network of Q&A communities. The largest of the communities is Stack Overflow, which is for programmers. The community members can post questions and answers, receiving or losing points while doing so [22]. The asked and answered questions, the total amount of points an applicant has, and where they have been received from, are the main attributes that recruiters can gather information from on this website.

2.2.3 Methods of evaluation

Most of the people who are closely involved in IT recruitment, and especially to recruiting programmers, all share the common opinion that it is difficult to generalize. Evaluating of programmer's skills in general is a daunting task. Programming consists of a vast number of different types and languages and each of these has its own know-hows and preferred ways of doing things. Knowing all of these would be required for giving perfect evaluations regardless of the candidate and the job opening. Therefore, instead of trying to master all of the aspects of programming, the next paragraphs present some ground rules that a recruiter has to meet and methods that they and other examining parties can use when going through the list of potential candidates.

For an IT recruiter to excel in their work they need to obtain the necessary knowledge of the technologies that the company uses and the ability to make circumspect decisions [16]. The latter means that, as mentioned earlier, to be ahead of the competitor, having the confidence to skip a phase in the recruitment process can lead to snatching the candidate before rivalling companies do.

In order to do this, a recruiter has to know the overview of the company's system and its architecture and preferably also some basics of how it is actually built. The main attribute that makes a person an advanced level IT recruiter is being interested in the industry in general. Being active and capable of self educating are also appreciated because the industry is ever evolving. Also, without knowing the terminology, being surrounded by acronyms will result in misunderstandings and will frustrate the recruiter. [16]

As in any job opening, résumés and application letters are a part of the evaluation. However, it is important not to hire the best résumé but the best candidate. A well executed evaluation model provides as reliable a result as possible by considering the candidate's knowledge, skills and abilities, accomplishing a realistic general view. [23]

When grading the potential candidates, trying to find evidence proving knowledge and capability is the most effective and trustworthy way of approach [6, pp. 6-7]. While it is easy to see the quantity of code a programmer has written, it is incredibly difficult for an inexperienced programmer, let alone a recruiter, to piece together the level of skill required to write that code, therefore the person making the final evaluations should be someone who possesses enough knowledge to make reliable deductions. For instance, programming of a complicated script might demand a lot of thought and time to be spent while, in the end, would only result in a few lines of code. On the other end of the spectrum, writing markup languages such as HyperText Markup Language (HTML) results in large quantities of code without requiring much thought. A candidate without any proof of skill should, however, never be considered a potential hire.

Along with experience, it is important to define the applicant's immediate and capability tools to figure out the true potential and benefit. Immediate tools are the mediums that a person obtains and capability tools are strengths that person could potentially have. To further clarify, the methods and construction means needed to build a software are immediate tools. Capability tools are the meta-tools which allow identifying, learning and innovating, for example to see when it is necessary to construct new immediate tools needed to solve a new set of goals. Capability tools include the ability to learn, master and teach the following:

- New construction methods
- New programming languages
- New techniques
- New platforms
- New software domains, such as embedded systems or cloud services
- Recognizing the need to innovate
- Innovating and building new tools
- Collaborating with experts in disparate roles and domains [6, p. 10]

Capability tools can be inferred from the applicant's personality and social- and cognitive learning skills. An experienced recruiter with the right set of tools, such as tests specifically tailored for the purpose, should be able to map out the candidate's capability tools. An example of such a combination could contain tests targeting personality, cognitive learning and performance under pressure. There also exist online assessment tests that are constructed for the purposes of determining applicant's capability tools [24]. After regarding these, some applicant may appear as a much more preferable hire than the others even though their previous experience would indicate that their abilities are at the same level.

Interviewing, either physical or via telephone, is a critical part of the evaluation model as it is generally the part of the model in which the recruiter can expect instantaneous answers from the applicant. The role of interviewer is often seen as easy, but there are a handful of reasons why interviewing often fails that are usually caused by the interviewer — these include poor time management, conversing instead of interviewing, making early conclusions based on first impressions, using ineffective questions and lacking knowledge of the subject [25, pp. 5-12].

Interview is not just a conversation and those that are great at conversations are not necessary even remotely good interviewers. The interviewer is in charge of the meeting and therefore schedules how time is allocated during the interview, still, during a typical interview the interviewer tends to do the majority of the talking. The purpose of an interview is not to mingle with or befriend the applicant, or to sell the company or the position, but to gather objective information about the candidate. [25, pp. 5-7, p. 12]

The question palette is also often wrong. Use of ineffective questions, such as "Where do you want to be five years from now?", are commonly used without much thought. An effective question is one in which the interviewer knows the right answer to and one that should not leave room for any interpretation. Lastly, an interviewer without the knowledge required to make differentiations between the qualified and the unqualified candidates is not an adequate interviewer. [25, pp. 10-12]

An IT recruiters' verbal communication has to adapt to the situation. A candidate may, for instance, be introverted or otherwise communicating in a socially awkward manner.

The recruiter has to be able to communicate with such a candidate to do well in their job. As another example, an established programmer may have a hard time explaining their previous work to an inexperienced recruiter, resulting in awkward conversation. A more experienced recruiter might not understand the context fully but since they know the terminology and can understand some sentences here and there, they can still communicate with the candidate.

When the amount of candidates has been narrowed to just a few, candidates' abilities seem to be at a same level and the previously mentioned methods are not sufficient enough to make distinguishing differences between the candidates, using a code test is also a viable solution. Instead of using a single code test, it is also possible to increase the length of recruitment process and have multiple code tests with increasing difficulty and then progressively eliminate candidates after each test. The issue with this is that as the process gets lengthier some candidates may lose interest.

Another problem is that code tests have to be specific enough and tailored for a specific job description or programming language or else they are too unreliable [26, p. 3]. Structuring of code tests also require a lot of time, if constructed properly, and if they are language neutral, meaning that they are not limited to a certain programming language, going through the results can be a painstaking task. However, a functional code test needs to be done only once and in the end the benefits outweigh the resources spent. Section 2.4 goes into deeper detail on the theory behind code tests.

Typically when the company's HR-team, recruiters and IT team leaders are working together, they can raise the curiosity of programmers towards the company with an interesting job advert, removing the need to sell the company later on [6, p. 49, p. 59]. Also, another benefit of such a collaboration is that only then the necessary knowledge to build an effective team of programmers is established. Team leaders know programming and the team composition, while recruiters have the expertise of communicating and grading the candidates. Using the knowledge of these parties, it is possible to gather information about what tools and methods can be used to make the IT recruitment process as fluent and reliable as possible, therefore resulting in hiring the most beneficial candidates as effectively as possible.

2.3 Defining candidate’s skills

The importance of figuring out the candidate’s competence and skills cannot be dismissed. IT industry is incredibly competitive and creating new tools and refining the old for achieving a swifter evaluation model are valued highly — code tests being one concrete example of this, as they are efficient and accurate on giving feedback on the candidates’ skills when structured correctly.

There are people that are naturally very good at something. This might be misinterpreted as being skilled. All skills must be obtained via practise, however. When someone has a biological prowess to learning something easily, such as mathematics or languages, they are not actually skilled in learning that thing, they are talented. Skilled learning means that a person is using an array of tools and methods that make learning easier for them. [2, p. 4]

Most skills can be categorized into either of the two, domain-general or domain-specific skills. Domain-general skills are skills that could be used in more than a single task. These are much broader and a good example of such a skill could be social skills. Modern teaching theories emphasize that instead of teaching dividual subjects it would be much better to try to build a bridge between them and merge them into a larger context [27]. A good concrete example of this are subjects like mathematics and physics, as both of these try to explain how the world works and therefore they complement one another. There are some differences, but knowing physics definitely does not affect mathematical competence. The reason why domain-general knowledge is often appreciated more is that when skills get combined into a larger context of knowledge, those skills actually become a tool that can be used to adapt to various situations.

Most job descriptions require both domain-general and specific skills, but there are also those that do not. Carpenters for instance can excel in their job without having many other skills outside of their carving skills. Or to modernize the job description, there are companies that are looking for Java programmers for positions that only have a single requirement, being good at Java. When looking into a typical job description, however, skills from both domain-general and specific skills are usually sought after.

Skill can be measured. If a set of job applicants are provided with a region neutral test and all applicants are working under circumstances that are most favorable for their performance, the applicant with the most skill on the subject of the test should score the highest. Alternatively, if multiple candidates score the same highest score, there is usually someone who finishes the test the fastest.

However, a situation like this is not very likely to happen because it is very unlikely for all the candidates to be at their best performance. Therefore measuring skill legitimately with any test is actually incredibly difficult if there is any type of pressure involved, such as in a situation where the test would be a delimiting factor for being hired or not. What companies can use such a test for is just as an addition to the whole recruitment process.

2.3.1 Defining programming skills

Before trying to map out what the things are that programming skills could potentially consists of, it is necessary to define a programmer. A programmer is someone who writes computer software. In this case a computer could be any machine that can interpret binary code. A programmer's skills consists of domain-general and domain-specific skills. One such domain-specific skill is programming skills. These are "the skills required to write a program so that data may be processed by a computer" [28].

In the list below, there are some criteria that make a good programmer:

- Good knowledge of at least one programming language
- Understanding of the underlying systems
- Effective use of command line tools
- Ability to debug
- Capable of testing the code
- Interest in current trends and future technologies
- Motivated to learn the new
- Co-operative, good at such things as team work and communication
- Obtains problem solving skills [29, slide 8.]

Items such as "Good knowledge of at least one programming language" and "Capable of testing the code" are examples of domain-specific skills that correlate to programming skill. Linus Torvalds describes the difference between good and bad programmers as the following: “Bad programmers worry about the code. Good programmers worry about data structures and their relationships” [30] and as mentioned before, once the domain-specific skills merge into a larger context, real knowledge is gained.

A similar kind of list for a person with good programming skills in web development could be something like the following:

- Writes working and efficient code that is readable
- Uses algorithms effectively
- Makes sure that the code is as secure and safe from cyber-attacks as possible
- Can create and maintain large projects with smart directory solutions
- Knows the theory and can apply it to needs (List mine)

Defining of general programming skills would result in conflicts, some attribute may overrule another and some of the listed items may not apply for specific programming languages or job descriptions at all. That is why the only reasonable approach is narrowing the context down to a specific type of programming, such as automation, mathematics or web development.

2.3.2 Evaluating the suitability of a candidate

There are some attributes to benchmark when evaluating and grading a candidate. In figure 4 some of the most important characteristics that need to be thought of are presented. Excluding medical health, the attributes in figure 4 are relatively easy for a recruiter to figure out.

There exist tailored tests that aid in the HR process, these tests often consist of some psychological, functional characteristics and scientific parts and the rest the recruiter has to determine from portfolios, interviews and other communication. For instance, obtained skills could be estimated with an assessment test: a test could approximate the candidate’s skill to allocate time accordingly, by providing some platform where there are multiple tasks to be done but some have higher priority than others.

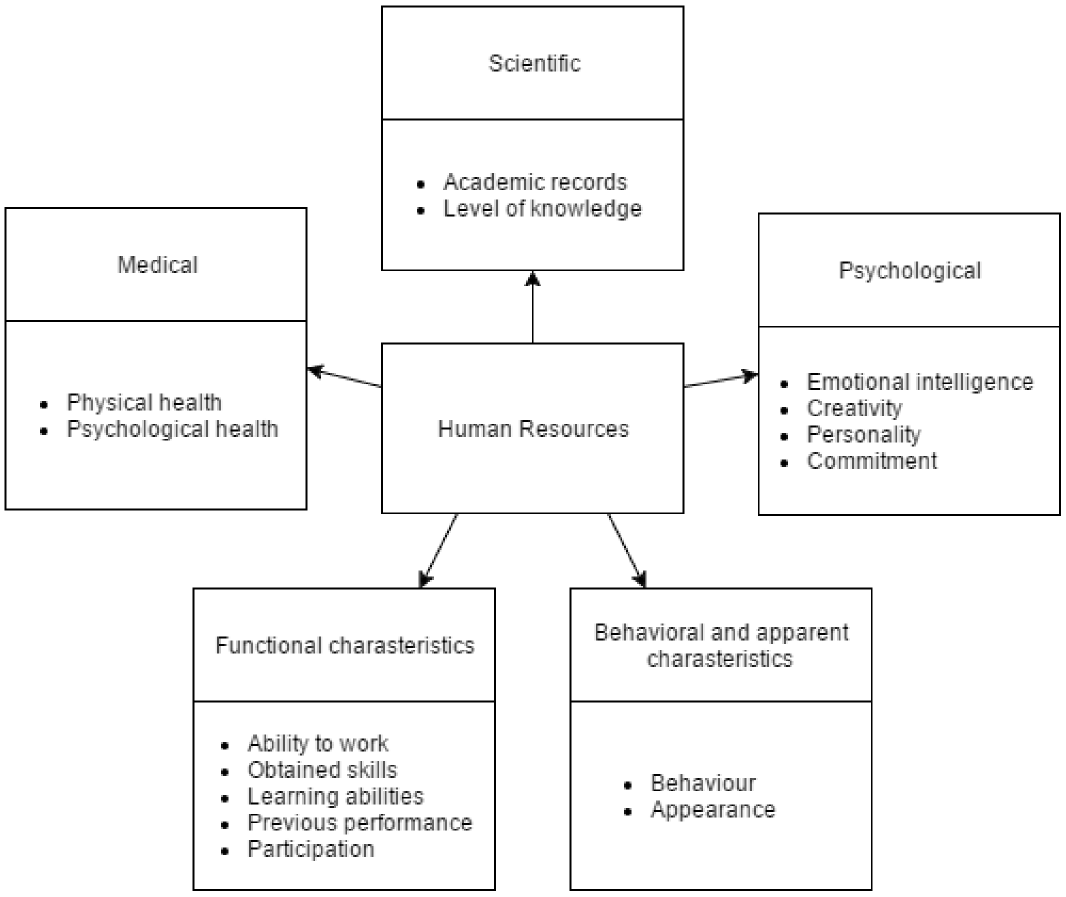


Figure 4: Attributes to benchmark while screening candidates. (Modified from Nobari (2012) [31])

When companies are trying to find the most suitable candidate for a job opening, they will need to pinpoint the important attributes and skills that they are looking for and then execute a plan for evaluating applicants. One way of approach is to make a list containing both domain-general and domain-specific skills and use it in the initial elimination process and later as a tool for grading candidates and finding the most suitable one.

The following is a list of requirements that one Finnish company, specialized in providing integrated retail and supply chain planning solutions, used in one of their System and Network Administrator job openings:

In this role your responsibilities will include:

- Deploy, maintain, configure, upgrade and retire servers and applications in accordance with company standards.
- Manage firewalls and network devices.
- Continually monitor network and server status using monitoring systems and react upon alarms.
- Troubleshoot applications and system problems.

- Participate in IT infrastructure development projects to support day to day operations and projects; investigate new technologies.

We value the following skills in our ideal candidates:
- Prior professional experience of Linux System Administration
- Prior experience with infrastructure hardware management and troubleshooting.
- You master scripting (Bash, Python, etc.)
- You are not afraid of responsibility and can handle multitasking
- You are fluent in both verbal and written English
- You have a good attitude, and you are fun to work with!

We consider as an advantage:
- Knowledge of orchestration tools like Ansible or Docker
- Experience with AWS or monitoring systems administration is also seen as an advantage
- Fluency in Finnish, Swedish, German, French or Italian [32]

In the previous example, the first section consists of the major themes and is useful when the recruiter has found multiple suitable candidates and differentiating has become difficult. The second section is what is used to execute most of the eliminating consisting of both domain-general and specific skills. Time periods such as “prior experience” are used to secure that candidates have the competence to manage those areas of the job. There are some attributes on the list that have to do with one’s capability-tools and in many cases those are weighted as the most valuable. Last section consists of domain-specific skills that can be used to make separations between otherwise equally capable candidates.

If the recruiting company is uncertain of how advanced the candidates are in programming there is always the option of using a code test or multiple tests. Programming tests are becoming a crucial part of the IT recruitment process, because when built right they can give quick and accurate information about the general knowledge of the candidate [33]. There are a few approaches to code tests but to build a language neutral code test, a test that is not specifically for a certain programming language and can be used in several job openings, the use of pseudo code, a language that is often used when teaching the basics of thinking as a programmer, is recommended. That is due to its nature of not favouring anyone and abstracting away the details and complexities of individual languages, enabling the candidate to purely focus on the problem at hand. An example of an aptitude test build with pseudo code is the Berger Aptitude for Programming Test (B-APT), presented in more detail in section 2.4.

There are also some companies that are specialized in code test platforms that are easy to use, sometimes even containing existing code tests to choose from. Some of these platforms are very advanced, providing real time screen sharing, direct contacting and advanced tools for ranking the candidates. The price tag on these platforms is usually quite low, considering that recruitment in general is quite expensive. For instance, Codility's cheapest model costs $159/month but is limited to just fifteen test attempts [34]. For companies that are recruiting on frequent basis but do not have a lot of capital, the more realistic option is using some platform that simply uses multiple choice questions instead, therefore making it a lot cheaper still. An example of such a platform is ProProfs, the platform used for code tests in Coding Factory 2017 [35].

2.4 The theory of code tests

Code tests are a part of technical assessment, and their use enables automating the otherwise somewhat manual recruitment process [33]. Typically they are built either as a quiz, offering options for the applicant to choose from, or in a format that presents a programming related question and a field in which the candidate writes a code block that tries to solve the described problem.

Code tests can either be online or offline, and range from a simple pen and paper test, in a closed environment, to a test in which the use of web is advised by the organization. A test typically consist of essays, producing code, interpreting code, purely theoretical questions or of just one of the previously listed question types.

In theory, the more items a test has, the more reliable it is, because then mistakes and misunderstandings do not have as much impact on the score [26, p. 3]. However, repetition should be avoided and the questions should be divided into sections to keep the interest of the applicant [26, p. 3]. As mentioned, code tests can be constructed by the recruiting organization or bought from a ready made platform. The tests also have to be considered according to what results they aim to achieve, whether the goal is to screen candidates, to support technical interviews or to build a more positive candidate experience [33]. This usually determines whether the use of Internet is allowed.

Most commonly, the code tests are online, the reason being that when a programmer is working on a project, they typically have access to the web, which imitates a real work place scenario. Providing a candidate with a test in a natural environment is preferable in verifying their capabilities [36]. The components of which the test can consist of should be based on the job description, meaning that each of the questions should be picked based on how they reflect the job. Code test providers offer assessment tests that are directly targeted to languages, frameworks and general knowledge, such as of architectures and systems that a program is built on [34].

The increased use of code tests has raised some discussion and resistance in the IT industry too, especially from the side of applicants. As code tests require time to answer to, they become incredibly arduous in large quantities, such as when applying for multiple different positions. There are advanced programmers that feel that these tests are an insult to them. If there is no guarantee for being hired, why would any senior programmer waste time proving their knowledge. One might thus ask what real additional benefit does a code test bring to regular interviews in these cases? As the question difficulty increases, the time it takes to answer one does as well, therefore for seniors the code tests can become an extensive burden to do properly. Another question that has been brought up is their trustworthiness. A well formed code test provides valid results of the applicant's current level of competence but it does not measure how they adapt to changing environments, such as new frameworks. [37] [38]

2.4.1 Aptitude tests

All code tests are aptitude tests, structured in a systematic manner and increasingly administered on a computer. They are used to evaluate how people manage and perform on tasks or react to different situations. The systematic structure yields objective results. Aptitude test consist of standardised methods for administration and scoring, achieving quantified results and comparability to the performance of other people who have taken the same test. [39]

Aptitude tests for computing jobs can be roughly divided into three different types:

1. A standard battery of tests estimating proficiencies such as numerical, logical and non-verbal reasoning which are necessities in the computer industry jobs.

2. A hybrid test consisting of items that ask for logical rationalization, numerical problem solving, pattern recognition, meticulousness and competency to follow complicated procedures. Both test types, this and the one before, do not require any actual knowledge of programming.
3. A programming test containing pseudo code or code from a real programming language, control structures (e.g. loops), look-up tables, sets, arrays, booleans, recursion and other programming structures. These tests can be used to determine the expertise of experienced programmers. [40]

One well known, although deprecated, code test is the Berger Aptitude for Programming Test Advanced Form (B-APT AF). In order for the candidate to score high in the test, they are required to know the basics and obtain some of the more complex skill areas needed in computing environments. The test is a job-related tutorial test and the language used in the test is custom-made for just that specific test with a resemblance to pseudo code. The examinee of the test writes the questions themselves, taking in account the program's specification, and also makes the instructions for a hypothetical computer. To make the test more demanding, a set of language rules can be applied, increasing the complexity of the practical situations. The amount of questions is 25 and the time frame is capped to two hours. [41]

The tests are only a part of the overall assessment procedure, as mentioned. Employers use them alongside interviews, academic results, application forms, recommendations and other selection methods. There does not exist a perfect test, as mentioned in section 2.3 of the thesis, and candidate's disabilities and the test setting can affect the test's trustworthiness — dyslexia being a good example that would affect the validity of the result received from a code test. If a time period is static and cannot be modified, a capable candidate may score poorly due to the time limitation. [39]

2.4.2 Information derived from time limits

The tests used in candidate screening typically have a time limit. This varies, depending on the purpose of the test and its difficulty. Time limit can be utilised as a source of information and in eliminating applicants, meaning that it can be used to determine if an applicant performs well under pressure, how they adapt to stress and what their abilities are in allocating time for each task at hand [42].

Time limit also provides knowledge about the candidate's interest in the company. If a more flexible time frame is given, such as days or weeks, some candidates may finish the code test quickly while others do not finish it at all, which reveals their motivation towards the company [16].

2.5 The Coding Factory 2017

Before presenting the material gathered from the Coding Factory 2017 event and using it to assist in determining how well the candidate evaluation model achieved its mission, it is required to define the purposes of the event. The main intent was to provide the companies with a set of suitable candidates while making aTalent Recruiting and the client companies more recognizable and improving their brand image. Each of the companies had its specific field of expertise, influencing the pool of candidates selected to participate.

To achieve the foremost important goal, finding of suitable candidates, a set of evaluation tools and methods were chosen based on their effectiveness and time to value ratio. The outcome of this evaluation resulted in choosing two participation methods; allowing applicants to either provide a résumé and other documentation or to do a code test.

The candidates would fill an application form to participate in the event, consisting of basic details, such as the candidates' contact information, and their preferred method of participation, either by résumés or by taking a test. The team I worked in chose that we would use three different type of tests, Java, JavaScript and a Global test. The reason for choosing three different tests, instead of just one, was to achieve as broad a candidate pool as possible and to prevent discriminating those candidates that did not possess knowledge of a single programming language.

The companies in the event were IT consulting firms and other firms that were searching for programmers. The event itself consisted of workshops and each company would create an application that would be used in their workshop during the event. Each company constructed their workshop for their field of expertise in order to seek the candidate that would best fit the profile(s) that they were searching for.

The differences of the companies' needs resulted in that the code tests had to be broad as well, consisting of multiple different parts of programming and not just one specific type of programming. However, all of the tests had some questions related to web development, as all the client companies were working in this domain. It also determined which domain-general and domain-specific skills, explained earlier in section 2.3, were highlighted as valuable while grading the candidates.

3 Methods and materials

3.1 Research and research questions

Recruitment industry is enormous and dividing it into smaller units, such as by using education and the job opening's requirements, along with dismantling of each individual process phase and describing its purpose, ease in understanding the industry in general. Recruitment process and its evaluating methods and techniques vary depending on the occasion and encapsulating the purpose of these refine how recruitment is perceived. Understanding the general view clarifies recruitment and is a precondition for development. The focus of this paper is to weight the value of one of these methods, code tests, and to analyze how test questions are structured and on what basis they are chosen. The research questions that I aim to answer are:

- What can be achieved using a code test?
- What are the differences between a functional question and one that is poorly made?
- How to construct an efficient code test for an event that could be used for evaluating large quantities of candidates?

To answer these questions, I compare the amount of applicants of Coding Factory 2016 to this year's event, gather existing questions from code test platforms, interview an event participant and reformat data gathered from the code test results, a byproduct of Coding Factory 2017. The contents of these I describe and evaluate later on in the research.

3.2 Execution strategy of the research

In order to find answers to the previously presented questions I use both quantitative and qualitative strategies. The goal is to track what improvements could be made on the code tests utilised in Coding Factory 2017's evaluation model, as well as to examine code tests in general to ascertain their benefits and downsides. This would result in directly increasing the knowledge of structuring functional code tests and more effective evaluation pipes. Another objective of this approach is that the end product could be easily modified for some other needs, such as in providing basic guidelines for recruiters that they could use for improving their current technical interview questions.

The material was gathered from the previous year's Coding Factory event, multiple online code test platforms[1], the Coding Factory 2017 code tests' results[2] and by interviewing a single participant via phone[3]. The phone interview consisted of questions related to the candidate evaluation pipe and the code tests. The interview questions[4] were chosen based on their ability to show flaws in the communication between the event organization and the applicant, as well as to present the issues in the code tests platform and the questions themselves. Details on the interview can be found in section 3.3.1. Its results will be evaluated to conclude whether the use of a code test improved the evaluation model compared to the year prior.

The restrictions of the material are that the data received from the point of view of the candidate is scarce, affecting its soundness. Another lack that this material has is directly caused by the code test platforms being reserved in sharing their questions to the public. Coding Factory documentation was also not well formatted, and a lot of the material had to be reformulated to improve its readability, which required a lot of resources.

For the evaluation of each code test question's quality, I use the standards developed by the National Center for Research on Evaluation, Standards, and Student Testing (CRESST). I try to systematically pinpoint the best and the worst questions out of the questions gathered from various sources for the purposes of further analysis using CRESST's criteria, explained in section 3.3.2. To prevent narrowing the scope of research only to question level, I harness SWOT analysis tool for the needs of this thesis. SWOT serves in revealing more on the functionality of the used evaluation model and in achieving a deeper knowledge of how beneficial the use of code tests is, as well as in screening how it fits into the whole evaluation process. All of these methods provide a different point of view and should result in a broad knowledge of what would be the optimal structure of a code test and when to use it, as well as reveal its usefulness as a part of the candidate evaluation model for an event such as Coding Factory.

[1]Code tests platforms' data can be seen in appendix 4.
[2]Results of Coding Factory 2017 tests' are in appendix 3.
[3]Questions and answers of the interview are in appendices 1. and 2.
[4]The interview questions are presented in section 3.3.1. but also in appendix 1.

3.3 Methods

3.3.1 Interviewing

The questions used in the phone interview were chosen regarding the weaknesses and strengths of using a code test as a method of participation. The Coding Factory 2016 candidate evaluation model did not include a code test, therefore, the questions were aimed to reveal how a candidate experienced the addition of a new way of participation. In comparison to the previous year's event, the amount of candidates substantially increased, from 139 to 263, and my initial intuition was that this would have been a direct cause of the new applying method.

I decided that I would make some of the interview questions so that they could yield either "yes" or "no" answers, but could be easily followed up by asking the interviewee to justify their answer, therefore providing further information. The spoken language during the interview was Finnish, and the questions used in the interview are roughly translated and listed below, as are the answers presented later on.

1. How did the entire candidate evaluation process feel like?
   a) Did you feel like the options given (resume + CV / code test) were sufficient and functional?
   b) Was the process humane, did the process and communication with the organizer feel distant?
   c) Would some other route for applying have been easier or functional (for example interview before the event)?
2. Regarding your current capabilities, did you feel that you could show the level of your knowledge and skills during the applying process?
3. Was the process too complex or long?
4. How would you describe your application experience in comparison to the ones you have had before?
5. Did you feel that the application process was more demanding that a regular one?
6. Before choosing how to apply, did you put thought into which applying method would yield the best chances of getting to participate in the event?

The time allocated for each question I planned ahead, deciding that a maximum amount of time for each question would be five minutes, making the interview last at most half an hour. First one of the questions had sub-questions in it, therefore some of the other question I devised so that they would return shorter answers.

Before starting the interview, I introduced myself and asked the respondent if they were in an environment where they would be able to focus purely on the interview, and not have some other business to attend to that would affect their answers. After this, I clearly described the reason of the interview to ease them into the correct mindset. Then I started asking questions, and after each answer, I asked a follow-up question that was about a specific part of each answer to get further details behind the answer.

Overall, I spent about one fifth of the time talking and the rest listening. The interviewee did provide throughout answers for each question, and the time frame I had provided for the interview was more than sufficient to fulfill the interview's purpose.

3.3.2 CRESST's criteria

To evaluate the technical quality of a test, CRESST provides a set of criteria that could be used to assess each question, and I applied them on the questions gathered from code test platforms. What the criteria is not meant for is evaluating the quality of the test as a whole. In general, all test items should appraise achievement of instructional objectives, weight the important aspects of the subject, accurately mirror the emphasis placed on important parts of instruction, measure participants knowledge on an appropriate level and vary in difficulty [26, p. 6]. This the criteria does not aim to achieve, but to simply define how to establishing the technical quality of a single question. The criteria defined by CRESST consists of seven areas:

1. cognitive complexity
2. content quality
3. meaningfulness
4. language appropriateness
5. transfer and generalizability
6. fairness
7. reliability [26, pp. 5-10]

Cognitive complexity is used to rank a question according to its difficulty, and there are six levels in the *cognitive complexity* scale: knowledge, comprehension, application, analysis, synthesis and evaluation. Knowledge questions require the least amount of know-how, usually used to measure candidate's memory rather than deeper understanding of the subject. A sample knowledge question in programming is displayed in example 1.

> Which of the following jQuery method checks if event.stopPropagation() was ever called on this event object?

Example 1: Knowledge (Copied from ProgrammingSkills (2017) (appendix 4.)).

The aim of this question is to just measure a candidate's knowledge of the jQuery library and does not require cognitive thinking. An upgrade to a question of this level is a comprehension question. A solid answer to a comprehension question, rather than asking for knowledge, asks for deeper understanding of information, and usually includes restating of something. A programming question that requires comprehension could be as in example 2.

> List three reasons for using jQuery library rather than pure JavaScript.

Example 2: Comprehension (Example mine).

While being a relatively easy question, it still asks for more than knowledge of a single subject. A proper answer to a comprehension question includes an explanation and some thought from the candidate, making it more effective than a knowledge question.

The next level of *cognitive complexity* is application. When the candidate is forced to first identify a problem and then solve it, understanding of general rules, methods and principles is needed. There are plenty of examples of application level questions in programming, similar to the one in example 3.

> Fix the bug.

```
<script>
function average(a, b) {
  return a + b / 2;
}
console.log(average(2, 1));
</script>
```

Example 3: Application (Copied from TestDome (2017) (appendix 4.)).

The question is easy to anyone with any mathematical and programming competence, but to correctly answer the question a candidate has to possess the skills to apply theory to practise, which is considered considerably more difficult than comprehension or knowledge.

Analysis is one notch more demanding and it differs from application in question context. While application level questions revolve more around broad themes and acknowledgements, the analysis questions present material that the question is driven from. An example question in programming could be similar to the one in example 4.

> You are building an application for purchasing flight tickets online for an airport. Structure a functional relational database model that could be used in the flight reservation system. The minimal requirements for the system are: details about upcoming flights, such as flight number, schedule, route, passenger capacity and airline, and the purchaser's details, such as surname and other basic details. Any additional fields are considered beneficial, but should be stored in a sensible manner.

Example 4: Analysis (Adapted from Exove (2016)).

To properly answer the question, a candidate has to analyse the material they are presented with and then construe the answer so that it achieves the goal accurately and in a sensible manner.

The subsequent difficulty level is synthesis which emphasises discovering and creating of new connections, patterns, perspectives or generalizations. In practise, it asks for combining ideas to form a new whole. In code tests, a sample question of this level of difficulty could hence resemble the question in example 5.

> What would you add to the list of requirements for this job opening?
>
> - Can apply programming theories to practise.
> - Knowledge of the following programming languages: JavaScript and Python.
> - Previous experience in test-driven development.
> - ...

Example 5: Synthesis (Example mine).

The candidate is offered a set of rules but has to find the relation between each item, and then further improve on the list. This demands for higher order comprehension of the subject and is therefore more demanding a question than any of the previously presented.

Last level of *cognitive complexity* is evaluation, evidence and reasoned arguments being in the centre of it. These ask for judgement of proposals and how they accomplish their purpose. When associated with programming, a question could appear like the one in example 6.

> What priority would you give to the items in the previous list of requirements, justify your answer.

Example 6: Evaluation (Example mine).

The answer to a question of this level is not obvious and opinions and controversies are often present. Basically, the candidate who can reason their answer the best should be graded the highest.

The *cognitive complexity* criteria also has its issues when adapted to code tests. One of these is in making clear separations between some *cognitive complexity* levels, such as application and synthesis and comprehension and analysis. The difference of application and synthesis level questions I used to categorize the questions with was determining if a question required implementing of new functionalities or, instead, improving old ones, as on the synthesis level the fixing of existing structures and code blocks is a precondition. For comprehension and analysis the division was done depending on whether the questions asked for only identifying of the material presented or making further deductions from it. The reason why these divisions are an issue is that the CRESST criteria states that, for example, application questions should always be less demanding than synthesis questions, which this division does not enable, as sometimes implementing of new functionalities is more difficult than reconstructing old ones.

Cognitive complexity determines the basis of the question, that is, how much the candidate is asked to invest their time and resources to the question. The other six areas of CRESST's criteria are to evaluate the importance and quality of a question in other ways. The *content quality* aims to estimate the relevance of the question. If the code test is meant for a Front-end programmer position, is it reasonable to ask questions related to Back-end?

Meaningfulness approximates the importance of a question, that is, how much can be gained from answering the question and is it worth the time. If all items are graded the same and they are each worth the same, is it reasonable to invest time to an essay question? This gets highlighted especially in those cases where the time frame given is not sufficient to complete all of the questions in the test.

Yet along with *cognitive complexity* and *meaningfulness* is *language appropriateness*. Is the question clear or is there room for misconceptions? For instance, in the application level question in example 3, should the purpose of the function be also defined or does the function name give enough identification on what it is supposed to do?

The next criteria defined by CRESST is *transfer and generalizability*, the setting of the test or question. How well the test situation resembles the actual work scenario? For instance, as mentioned, tests that mimic programmer's work environment as much as possible can be considered more reliable than ones that do not. Same goes for a single question: if a question is purely theoretical and the likelihood of facing a real situation where it might pop up is close to none, is it well-grounded to ask?

Fairness is the next criteria specified by CRESST. A question should not take sides or give advantage to anyone, irrelevant of the subject. As a base rule, a question should present clearly the formulated task and another question should not aid in answering another. The examples previously presented in *cognitive complexity* level synthesis and evaluation I chose based on identifying and registering this issue. If a candidate was unable to answer the first, the second question would result in a fail as well.

The last criteria is *reliability*. A good question is one that cannot be guessed and requires understanding of the context. Multiple choice questions are often scorned as multiple guess questions. Therefore, they should be designed so that the correct answer cannot be deducted from the options. One approach is to make all items resemble one another and so that the right answer is not so obvious for even the examinees that have an adequate understanding of the subject.

To assess the benefits of the use of code tests I used SWOT analysis, explained in the next section 3.3.3.

3.3.3 SWOT analysis

SWOT is an assessment tool that allows exploring both the internal and external factors that may influence, in this case, the model's performance and effectiveness. The acronym stands for: Strengths, Weaknesses, Opportunities and Threats. Originally the

SWOT method was developed for the needs of business and industry, such as to screen the competitors of a company, but it can be adapted to work on multiple occasions, for instance in education. A SWOT analysis works as a guide when planning on improving the current model. It takes in consideration the model's inner strengths and weaknesses (S-W) and outer opportunities and threats (O-T). A SWOT analysis is useful for exploring new solutions to problems, making deductions about which path to take in the developing process of the model and in clarifying which parts of the model could be improved on. It is also useful when making prioritization on each model component. The elements of a SWOT analysis are to reveal which components work together and to identify potential problems that may need to be addressed. In order to achieve this, listing of internal and external factors is required. [43, p. 18]

The major reason why I decided on using a SWOT analysis as a tool for improving on the model is that it enables seeing the evaluation process in both smaller components and as a whole. There are two groups of people involved in the recruitment process, the evaluators and the applicants. The SWOT analysis' internal parts, the Strengths and the Weaknesses, can be used for making improvements on how the evaluators could better their work, and the external Opportunities and Threats are useful for seeing the model's marketing potential, as well as to screen how the candidates see the applying process.

I constructed a table for code test evaluation model component that is based on SWOT analysis, offering general-purpose identifying and a more specific view of the importance of it. What I aim to reveal using this method is the value of a code test as a part of the evaluation model and also to examine how each party sees code tests. The information from SWOT analysis should help in disentangling the thesis' question related to the general use of code tests.

3.4 Materials

3.4.1 The point of view of the candidate

In order to collect some data from the candidate's perspective of the evaluation model used in Coding Factory 2017 event, I structured a set of questions that were asked in a phone interview. Due to the fact that the event was already held, and there was not much to be gained from the participant’s point of view, I restricted the qualitative research to a single candidate. The information gained can only be considered subjective for that matter and consequently it is only used alongside the data that is available for making comparisons from Coding Factory 2016.

The data from the interview I transcribed to a Word document.[5] The contents of this document are the only gathered information that was received from the participants regarding the evaluation process utilised in the event.

3.4.2 The questions in code tests

When searching for examples of existing code test questions for this research, I faced an issue regarding existing examples. The high-end platforms that are more well thought of and trusted are quite reserved in providing examples of their question palette, which resulted in the material gained from these sources being quite limited, more so than initially expected. Some of these, such as Codility, even state that the example questions are not to be copied to any instance, not even for research purposes [44, section 9]. Luckily not all the platforms are so strict with their policy and I managed to gather a relatively well-sized pool of 39 questions. From these I divided them according to their cognitive complexity, a criterion standardized by CRESST.

Initially I stored all the material from various sources to a single Microsoft Word document, which can be seen in appendix 4, and for the reorganizing I made another Word document that is displayed in appendix 5. in which the questions were parsed according to their difficulty. The reasoning behind this was to estimate the current level that code test questions are and to recognize any patterns in which the questions are structured in. This information would be valuable when making improvements on code test questions in general.

[5]My translation of the interview transcript can be found in appendix 2.

3.4.3 Coding Factory 2017 data

Use of code tests as a path for participation to Coding Factory 2017 resulted in a large quantity of candidates applying using this method. Out of the total of 263 applicants, 109 chose to apply using a code test. The test results were initially stored to an online database, managed by the test platform ProProfs. The ProProfs platform was chosen due to its affordability, while having sufficient evaluation methods for the occasion. The result data stored was not formatted, making it difficult to make deductions from. My first intuition was to reorganize it so it could be used effectively. The ProProfs platform had a separate table for each of the tests, consisting of test reports and this data could then be converted to Excel format (.xlsx) and exported.

From the exported data I reconstructed tables in Microsoft Excel and for each test I created a separate table. I also composed one table that was to summarize the data and give an overview of candidates' performance. After this was done I reorganized each table in a descending order so that the candidates that scored the highest would be presented on top. Doing this allowed me to easily find out what the median score for each test was. The functions implemented in Microsoft Excel allowed me to calculate the average score and time spent doing each test.

The next step that I took, while re-ordering the data, was to find how many candidates were accepted using a code test and then to calculate the portion that this was of the total number of participants. I also calculated the average time that each candidate spent doing the test and how long each individual question in the questionnaire did take. By doing this I achieved to map out which questions were obviously too easy and which ones took considerably more time to answer to, making them too difficult or too time consuming, and therefore they should be left out or changed in some way in the future. The analysis and results of this data are explained in moderate detail in the next section 4.

4 Analysis and results

The results that are discussed in this chapter were gathered from Coding Factory 2017 code test data, the phone interview had with a participant and from applying CRESST criteria to existing questions from code test platforms. The conclusions driven from these are presented in chapter 6.

There are some clear patterns that can be seen recurring in individual code test questions on all difficulty levels, shown more specifically in section 4.3, and it was possible to model a schema that makes a clear distinction between a functional and a poorly made question. The questions that would benchmark the candidates' level of knowledge and obtained skills, characteristics that are typically assessed in the candidate evaluation as displayed in figure 4, would theoretically be graded the highest. Both the phone interview and the material in the next section are gathered from Coding Factory 2017, and the questions from the code test platforms are discussed in a more general way, as the results of these may also be put into the event's perspective. Considering that the material gathered was not quantitatively optimal, it shows that certain test providers clearly had a better understanding of structuring more demanding questions than the rest.

When it comes to choosing between a single code test or multiple code tests for events, the result is also clear. The amount of available pseudo code quizzes that I was able to find from code test platforms was none. Therefore, it appears that they are widely considered too inaccurate to use in any a assessment test for programming. This resulted in the deduction that, to achieve a broad pool of candidates for an event that is not restricted to a single language, there should be plenty of code test options to choose from, rather than a single pseudo code code test. However, there were a few questions available that could be used as a part of a code test for almost any programming language. The results of each individual assessment are discussed in further detail in the next sections.

4.1 The results of Coding Factory 2017's code test data

The code test data from the Coding Factory 2017 event was mainly used to approximate the usability of code tests as a part of event's applicant evaluation process. In figure 1 there was an example of a simplistic and efficient evaluation pipe, in many ways similar to the one utilized for Coding Factory 2017 event other than it did not include a code test. However, as mentioned previously, the increase on applicant amount might have been caused by the addition of having code tests as an applying method and it seems that this implementation did not affect the operability of the evaluation pipe drastically. Therefore, it is important to evaluate the benefits of including code tests into the recruitment process. The following figure 5 shows the distribution of applicants' test scores for each code test used.

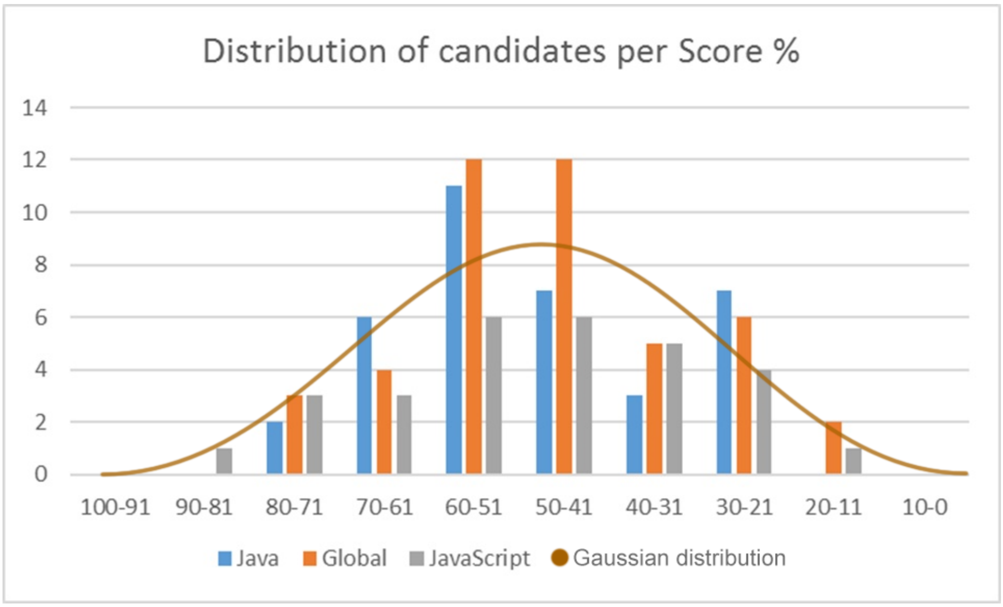


Figure 5: Distribution of candidates per score percentage.

From figure 5, it is possible to deduct that the JavaScript test used for Coding Factory 2017 applying process was structured in a more optimal manner than the others. If each code test is compared to an ideal situation, that being the Gaussian distribution, the test that resembles this the most is the optimal. That being said, the JavaScript test has most variance in comparison to amount of test-takers, making it the most effective and, therefore, trustworthy of the three tests used in Coding Factory 2017 evaluation model.

The Global test was requested the most, and therefore seems to be the most appealing of the three code tests that were available. The amount of applicants that did each test are displayed in figure 6. However, the difference in the number of Global, Java and JavaScript test-takers is not very substantial.

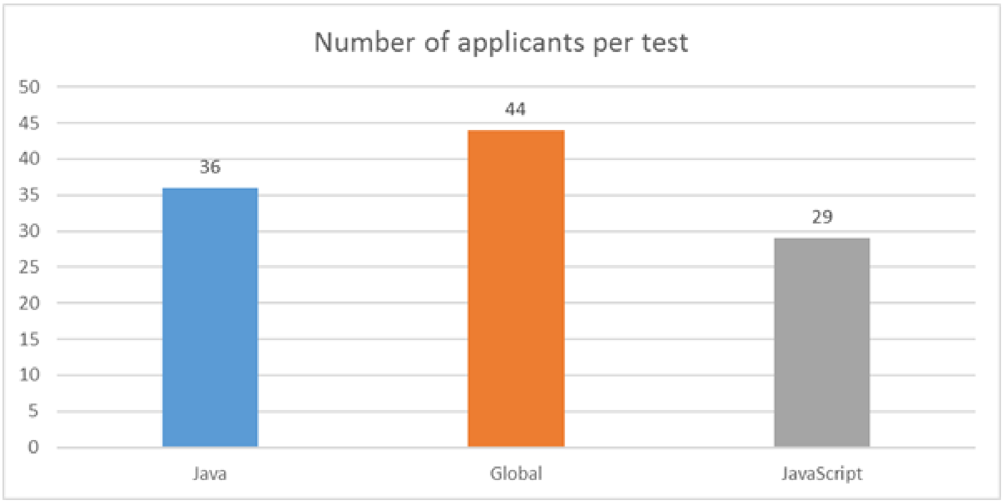


Figure 6: The number of applicants for each code test type.

Assuming that the applicants chose their test rationally, we can conclude that general knowledge was perceived as easier than domain-specific knowledge, while it is quite the opposite in reality when it comes to programming, as explained previously in section 2.3.1. Another option which could have had an effect on decision making, and could be the reason for this result, is if the Global test was marketed as easier than the others. The interview proved this contingency wrong though, as the interviewee clearly stated that “it did not seem like it made much of a difference for my chances which application method or test I chose”. Last option would be that as there are multiple different types of candidates – as explained in section 2.2.2 – some of them might simply choose the test by random.

4.2 The results of the phone interview

The questions used in the phone interview and the outcome of these can be seen in detail in the appendices 1 and 2. In this section, the most notable arguments are being presented to clarify the perspective of the candidate attentively.

As mentioned, in section 2.3.2, the biggest downfall of a recruitment process is its evaluation model being too complex. To assess this, a set of questions were targeted to map out the quality of the process used. The responded gave positive feedback to all of the questions related to the evaluation process, extolling that it was “straightforward and functional". The interviewee even noted that the process used in Coding Factory 2017 was "clear and easy", and that it effectively eliminated the less promising candidates from the ones that deserved to participate:

> "Based on the conversations I had had with the other participants I felt that all the candidates were well-suited and so the process was clearly functional."

In order to gain further information about modifying the evaluation process into a more humane one, I asked if they would prefer a short phone interview over the other applying routes. The interviewee responded with:

> "I do not think any changes are warranted (to the evaluation process). If I am required to express my motivation, it is easier to do so in a written form."

This would imply that increasing the amount of human interaction is not necessarily a benefit, and that applicant friendly evaluation process can also be one where most of the communication is done in a digital form. If there were a more extensive pool of respondents, the answers might vary a lot. However, as not to discriminate against any candidates, providing an evaluation pipeline that does not measure the applicants' social skills would likely be preferable. As in the case of Coding Factory 2017, the preliminary candidate evaluation was to grade the applicant's programming capabilities, not people skills.

When asked if the interviewee felt that the application process was more demanding than a regular one, they answered:

> "On the contrary, I felt that it was less burdening, as it is easier for me to answer questions than to make guesses about what the recruiter is interested in knowing about me and trying to write something based on that into my application letter."

This would indicate that having the option of choosing a participation method would reduce the amount of stress an applicant feels during the process. This also clearly implies that the amount of applying methods increases the amount of applicants. The answer also verifies that application letters may be left out from the standard candidate evaluation pipe in IT job openings, as mentioned earlier in section 2.2.1.

When it comes to the questions used, the candidate stated that "one of the questions was ambiguous and it was not obvious how in-depth an answer was expected", therefore, at least one, if not multiple, questions should be modified or replaced. Also, the interviewee offered a solution to this issue: "there should be some restrictions, for instance with word count", which is a valid point, as it is one that would clarify the meaningfulness of the question, a criterion defined by CRESST.

As mentioned earlier, I tried to reason why Global code test was requested the most. To address this, I asked if the interviewee put thought into selecting their applying method:

> "No, I did not. I chose the applying method which seemed to give me the highest chance of getting picked with the lowest amount of effort. However, it did not seem like it made much of a difference for my chances which application method or test I chose."

This response is one that I would not make many deductions from. Most likely at least a few applicants did consider their odds when choosing the best way to score a participation ticket.

4.3 The results driven from CRESST criteria

Using the CRESST criteria on the questions gathered from code test platforms revealed that some of the questions used for the evaluation of applicants for Coding Factory were inefficient and provided only obscure results. As a summary, there were too many language specific and pedantic questions, a lot of knowledge questions the answers of which could be found from online sources easily, and not enough comprehension questions. However, due to the policy of the client company, only the material from code test platforms is discussed in further detail in this and the upcoming sections.

In the material collected from code test platforms there are twelve knowledge questions, ten comprehension questions, nine application questions, two analysis questions and three synthesis questions. Most of these questions are either from JavaScript or Java tests, as these programming languages are ones that were also used in the code tests for the Coding Factory 2017 event. Some of this material is divided according to the CRESST's cognitive complexity criterion in the next section 4.3.1. [6]

[6]Appendix 5. contains all the other code test platform questions parsed according to the cognitive com-

4.3.1 Cognitive complexity criterion

Certain factors often determine how difficult a question can be. The more demanding questions most commonly require writing of code, and are therefore only doable on platforms that offer some sort of editor for the applicant to use. In contrast, the easier questions that suit well into multiple choice quizzes have downsides if the test is done online.

Knowledge questions

Knowledge questions, as the least demanding level of questions, suffer the most from the use of internet. As mentioned, these questions ask for memory rather than understanding. Therefore, it is possible to find solutions to most of the knowledge questions relatively easily using search engines, such as Google. The following question in example 7 is a poor knowledge question.

Is jQuery a programming language?

Example 7: Poor (Copied from ProgrammingSkills (2017) (appendix 4.)).

Querying the previous question will yield an immediate correct answer from any search engine. Therefore, knowledge questions make the test untrustworthy if every question is a knowledge one. However, in the data I gathered there were a few exceptions too, such as the one in example 8.

How do you fetch the first span on the page, which has the class ”green”?

Example 8: Better (Copied from ProgrammingSkills (2017) (appendix 4.)).

It is still possible to find the answer online, but it is most likely more time consuming, therefore making the question in example 8 a better knowledge question than the one in example 7. In the material gathered, all knowledge questions ask for domain-specific skills and knowledge, explained in section 2.3, as they are purely focused on specific details of one programming language and not broader contexts. The strength of knowledge questions is that they are fast to answer to and easy to grade, suiting well to multiple choice tests, as they do not require any interpretation to be done by the inspector and may then be graded by an examiner with no programming capabilities. This means that, even an advanced IT recruiter could grade these questions if they possess the attributes explained in section 2.2.3.

plexity criterion.

Comprehension questions

Similar to knowledge questions, the comprehension questions function really well in multiple choice tests, as they do not require any programming to be done by the applicant. Similar to knowledge questions, these questions are fast to structure and grade. However, comprehension questions are substantially more difficult to find answers to from online sources, which make them more ideal to use in online tests. In example 9 there is a comprehension question that is difficult to find direct answer to from search engines.

What will be the output of the program?

```java
public class CommandArgs {
  public static void main(String [] args) {
    String s1 = args[1];
    String s2 = args[2];
    String s3 = args[3];
    String s4 = args[4];
    System.out.print(" args[2] = " + s2);
  }
}
// and the command-line invocation is
// > java CommandArgs 1 2 3 4
```

Example 9: Comprehension (Copied from TestDome (2017) (appendix 4.)).

As in example 9, comprehension questions are often tied to a single programming language and require adjustments to make them usable for tests in other programming languages, which takes assets. From the material gathered, there was not a single question that could be used without changes in other programming languages. Additionally, comprehension questions may feel rather repetitive and quite time consuming for the applicants to answer to, making them tiresome in large quantities.

Another remark from the gathered material is that there was not one comprehension question that was less effective than the knowledge questions in revealing programming capabilities of the applicant. In section 2.3.1, there was a acknowledgement that good programmers are able to write working code that is readable. To write code, one has to also be able to read it, and identifying and interpreting of existing code blocks is a must in order to develop new ones. Also, as mentioned in section 2.3, domain-general skills, the skills that have merged into a larger context require further understanding of the task at hand. The previous question asks for these skills as the applicant is forced to understand programming in practise too, and not just the theory of it. Therefore, I feel like use of comprehension questions, instead of knowledge questions, is recommendable in multiple choice tests.

Application questions

The strengths of application level questions are that they are relatively fast to structure and, as previously stated, prove the candidate's level of skill more accurately than the previously presented cognitive complexity levels, as the applicant is required to write code. Other benefits of using application level questions is that they can be structured to any difficulty, from easy to hard, and that they can be targeted to be strongly relevant to the job opening, as in example 10.

> A TrainComposition is built by attaching and detaching wagons from the left and the right sides.
>
> For example, if we start by attaching wagon 7 from the left followed by attaching wagon 13, again from the left, we get a composition of two wagons (13 and 7 from left to right). Now the first wagon that can be detached from the right is 7 and the first that can be detached from the left is 13.
>
> Implement a TrainComposition that models this problem.

Example 10: Application (Copied from TestDome (2017) (appendix 4.)).

The identifying of problems and finding of solutions are commonly asked capability tools – clarified in section 2.2.3 – for all programmers that have to work with data flow, making the example ideal for any programming position that demands these attributes. Another observation that can be made from the previous example is that it is language neutral, meaning that the question could be answered using any programming language. The weakness of application level questions is in the slowness of grading them. There are often more than one working solution to these questions, and therefore to grade them correctly, the evaluating has to be done by someone who knows programming well enough.

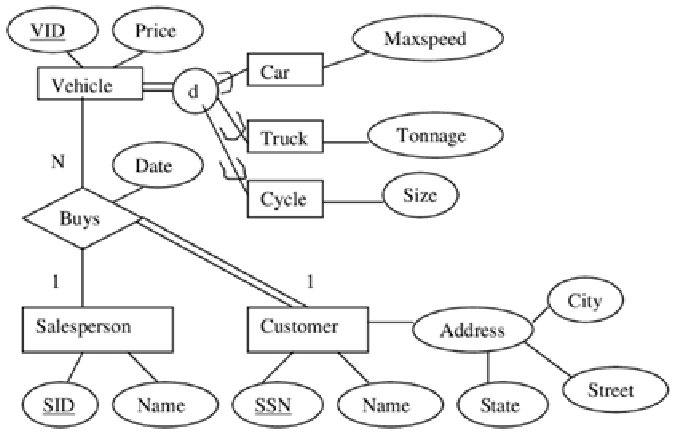
This perception signifies that, as the question difficulty increases, the test reviewer has to possess at least an equal level of comprehension of the subject as the applicant. Additionally, when the question is language neutral, effective reviewing requires domain-general skills because there are multiple solutions. As mentioned in section 2.3, when there is not a single task or approach, domain-general skills are required. To properly grade the previous example, the evaluator has to understand the question and its solution so thoroughly that they may, so to say, see behind the programme and understand each decision and the thought process that the applicant went through when solving the question.

Analysis questions

Finding of analysis questions proved to be difficult, as most programming related questions either ask for an output of a function or writing of new code blocks, as in the example questions used in the comprehension and application level. The only obvious sample questions that I could find were ones that present a figure, such as in example 11.

Consider the ER diagram below for the next question.

1. Which of the following statements is true?
   a) A Customer does not have to participate in the Buys relationship
   b) One Vehicle can be considered both a Car and a Truck
   e) A Customer buys a vehicle from one Salesperson
   f) A Customer is identified by the combination of Street, City and State



Example 11: Analysis (Adapted from docsity.com (2017) (appendix 4.)).

As can be seen from the question in example 11, the analysis questions adapt theory to practise really well, in this case requiring understanding of relational databases, making them effective for interpreting any data structures. The ideal use case for analysis questions is indeed in data-analysis and system architecture questions in which they work effectively. Another positive is that analysis questions bring variance to a test and may often also be used in multiple choice questions, which make grading of them quicker than application level questions. The biggest weakness of analysis questions is that they do not directly evaluate applicant's programming skills, but rather their understanding of the architecture. Also, they require a substantial amount of time to answer to, so the quantity of these questions has to be decided according to the time frame of the test, which can be hard to estimate beforehand.

Synthesis questions

The synthesis questions, as application questions, may be adapted to all difficulty levels. They are quite accurate in showing the applicant's programming skills, but are also time consuming to structure and grade because they cannot be used in multiple choice questions, as can be seen in example 12.

> Function appendChildren should add a new child div to each existing div. The divs should be decorated by calling decorateDiv.

```html
<!-- For example, after appendChildren is executed, the following
    divs: -->
<div id="a">
    <div id="b">
    </div>
</div>
<!-- should take the following form (assuming decorateDiv does
    nothing): -->
<div id="a">
    <div id="b">
        <div></div>
    </div>
    <div></div>
</div>
<!-- The code below should do the job, but for some reason it goes
    into an infinite loop. Fix the bugs. -->
```

Example 12: Synthesis (Adapted from TestDome (2017) (appendix 4.)).

What is also noticeable is that, unlike application questions, these questions are not language neutral, for the reason that there is an existing programme that needs to be altered. This means that they do not require as much domain-general skills to evaluate. The language restriction results in synthesis questions having the most strict use case and this affects their re-usability.

Evaluation questions

Theoretically, evaluation questions would measure an applicant's skills the best, as they are rather demanding to answer to and may show large differences between the candidates. However, what really stood out from the material gathered was that there were no evaluation level questions available. Opinions and reasoning simply do not suit assessment tests because they are only subjective. Subjective evaluation, in comparison to objective, is a threat, as the opinions of an examiner may affect the grading of answers equally. Other downsides are also present. These questions would be time consuming to grade and definitely not useful in multiple choice tests. Therefore, it is good to recognize that while they are an option, they should be considered thoroughly before implementing as a part of the test.

4.3.2 Other CRESST criteria

Now we turn to look at the other CRESST criteria: content quality, meaningfulness, language appropriateness, transfer and generalizability, fairness and lastly reliability. It is important to consider each of these attributes so that none of the properties that make a good question get overlooked.

To approximate the differences concerning the content quality of a question, meaning how well the question is related to the subject, I have selected a few samples, displayed in examples 13, 14, 15 and 16.

In example 13 there is a knowledge level question that could be a part of a code test for a Full-Stack JavaScript developer.

Which of the following is a server-side JavaScript object?
1. FileUpLoad
2. Function
3. File
4. Date

Example 13: Copied from ProgrammingSkills (2017).

In example 14 there is also a knowledge level question for a JavaScript code test. The concrete difference in comparison to the question in example 13 is that in this question the respondent is required to have some experience with jQuery, a JavaScript library.

Look at the following selector: $(”div”). What does it select?
1. The first div element
2. All div elements

Example 14: Copied from ProgrammingSkills (2017).

JavaScript is typically used as the logic of Front-End, and in a hypothetical situation, a test that would include the two questions, presented in examples 13 and 14, to measure the programming skills of the applicant, would be poorly structured. The question in example 13 asks for a very niche detail and the question in example 14 is not even a pure JavaScript question, as it requires knowledge of a JavaScript library and not the language itself. These questions would not be valid in a JavaScript code test, unless the job opening is for Full-Stack JavaScript developer that uses jQuery as a library.

The question in example 15 is of application level on the cognitive complexity scale and in it was found from a Java code test.

> A palindrome is a word that reads the same backward and forward. Write a function that checks if a given word is a palindrome. Character case should be ignored. For example, isPalindrome(”Deleveled”) should return true as character case should be ignored, resulting in ”deleveled”, which is a palindrome since it reads the same backward and forward.

Example 15: Copied from TestDome (2017).

Similar to the question in example 15, the question in example 16 is an application question that was discovered from a Java code test.

> The createProductCodeForm function is used to create a new form that accepts a product code from a user. The current version of the form contains the hint: ’The product code can be found on the label’. This hint is currently always visible to the user. Improve the form so that the hint is only rendered when the input element is the focused element.

Example 16: Copied from TestDome (2017).

The questions in examples 15 and 16 are ones chosen because they are structured so well that they are not language dependant but are still effective for weighting the applicant’s capabilities in Java.

In contrast, using the same examples to measure the meaningfulness of a question would yield opposite results. If each question was graded the same, the first two questions – in examples 13 and 14 – would be much more meaningful, because they are relatively quick to answer to. The other two questions – in examples 15 and 16 – are clearly more demanding to answer and are therefore ones that are not worth the candidate’s time. This would mean that in order to use the more complex questions, the points received for each question have to be thought thoroughly.

To approximate how well a question transfers to the actual job of a programmer, the questions in examples 13, 14, 15 and 16 are also good for. The palindrome question in example 15, is a often used one, and in reality there is probably never a case where a programmer has to write a function that checks if a word is a palindrome. However, this example and example 16 are a lot better for verifying the applicant's level of skill when it comes to programming in general, requiring understanding of a problem at hand and finding a solution for it.

The questions in examples 13 and 14 are ones that would never occur in an actual work environment, meaning that knowing what type of object to use, or what a certain selector selects is irrelevant for a programmer to know beforehand. These types of problems are ones that programmers use search engines for. "What object to use server-side in JavaScript" and "How to select all the divs from DOM" would be the search queries that were used to find answers to the questions in these examples in an actual workplace scenario.

When going through the material I found only one question that was clearly unfair. The sample questions in examples 17 and 18 include the aforesaid unfair question, and one that is relatively similar while still being fair.

> What is the correct syntax for Array Declaration?
> 1. var city = new Array("delhi", "agra", "akot", "aligarh");
> 2. var city = ["delhi", "agra", "akot", "Aligarh"];
> 3. Both 1. and 2.
> 4. None of them.

Example 17: Copied from ProgrammingSkills (2017).

> Which of the following are the ways to create a Javascript Object?
> 1. var obj = {};
> 2. var obj = new Object();
> 3. var obj = Object.create();
> 4. None of them.

Example 18: Copied from ProgrammingSkills (2017).

From the two questions in examples 17 and 18, example 17 is unfair while example 18 is not. There is not one "correct" syntax for declaring arrays as proposed in example 17. Array declaration may be done with both the ways presented in options 1 and 2, but using of the "new" keyword is considered bad practise, as it is slower in comparison to using the "[]" declaration. Therefore, the question itself is a trick question. Both options are technically correct, but the other option is preferred. In example 18, the question does not contain any obscure word, such as "correct", which results in making it more suitable when it comes to its fairness.

All questions in the gathered material were language appropriate, excluding small linguistic incoherences. In some cases word “object”, as in example 13, was lowercase and in others it was uppercase, such as in example 17. These mistakes are likewise seen in other keywords, such as with “class” and “array”. From the point of view of an applying programmer, these aberrations are disruptive and if these questions were used together in a test, the applicant might question the competence of the current programmers in the job providing organization.

To summarize, as the cognitive complexity of a question increases, so does the time it takes to grade them outside of analysis questions as these may additionally be utilised in multiple choice tests. Other types of questions that may be used in these tests are knowledge and comprehension. The knowledge and comprehension questions should take the biggest portion of these tests as analysis questions are more time consuming to answer to.

The other cognitive complexity levels; application, synthesis and evaluation only function in tests that provide an editor or some text field to write to. Not including evaluation, as there were no examples of these in the material gathered, these question levels are the most configurable ones, meaning that they can be adapted to any difficulty. Furthermore, they are best suited for evaluating the programming skills of an applicant. Especially application questions showed a lot of potential in them, as the same question may be used in multiple different programming languages.

Lastly, the other CRESST criteria proved that poor formatting of a question can lead to misconceptions and will decrease the reliability of results severely. Also, the meaningfulness criterion is one that has to be kept in mind at all times when composing a test, as even distribution of the points between each question is not always the right option to take.

4.4 SWOT analysis results

Strengths and Weaknesses (S-W)

As mentioned in section 2.2, e-recruitment, which code tests are usually part of, automates the recruitment process by providing instant results of the candidates' performance. There is almost no other field of work where a test could be used in a similar fashion as in IT job openings. For instance, to evaluate how good a salesperson is at their work, it would be practically impossible to construct a test suitable for this occasion. Programming is universally the same, and tests can be structured based on this fact to resemble real life programming work, while also remaining equal for all applicants, also enabling a more effective and efficient evaluation pipeline. The tests also ease in one problematic aspect of candidate evaluation, technical assessment, as the interviews and other commonly used techniques usually do not cover this accurately enough, referring back to section 2.4.

Some problems that do occur, when using code tests, are ones that all assessment test have. As mentioned in sections 2.2.3, 2.3 and 2.4.1, tests may also lengthen the evaluation process, and can never be equal to everyone, meaning that the stress or other factors involved in doing a test are always present, which may affect an applicant's performance. Other factors that have an effect on the soundness of a test are caused by the party providing the test. The questions in the test may be poorly structured or the test in itself might not be related to the subject. This will result to faulty data and favoring some candidates over others, which is against the CRESST's fairness criterion described in section 3.3.2.

When multiple applying methods are presented, another problem is faced. In the Coding Factory 2017, the percentage of candidates chosen based on their code test performance was 54%. When comparing to the total number of applicants, nearly every fourth candidate (24.77%) that did some test was accepted to participate in the event, making it a substantially more likelier way of scoring a participation ticket in comparison to applying using documents.[7]

This would indicate that equal grading of candidates becomes more difficult if the use of multiple applying methods is chosen. Two candidates may have almost alike programming background and if their portfolios are on the same level, should the one that has chosen to do a code test as their route to participation have a higher chance to participate or being hired than the one that has not. To address this problem, there is only one unambiguous solution and that is to only narrow the available evaluation methods to either documents or a code test.

[7]Further details can be seen in appendix 3.

These acknowledgements are listed in the internal parts (S-W) of the SWOT analysis' table 1, and may be compared to one another. The outcome of this comparison is that code tests are perceived as a way to lighten the amount of work that the recruiters have to do, while increasing the amount of work for developers.

Table 1: SWOT analysis for the use of code tests as a participation method for events.

Strengths	Weaknesses
• Automates the recruitment process. • Resembles real life programming work. • Enables effective and efficient grading of candidates. • Substantially eases technical assessment. • Results may by inspected without programming competence if there is only one right answer to each question. Such as in the case of multiple choice questions. • Gives concrete proof of knowledge. • Can be re-used any number of times and is not restricted to a single use case. • Yields immediate results of candidate's capabilities.	• Time consuming to structure properly. • May yield faulty data if poorly made. • Hard to balance each test to be equally difficult. • The test must be neutral so that it does not favor any particular type of applicant. • Objective evaluation of candidates becomes more difficult: how to make a differentiation between candidates if one has done a code test while the other has not?
Opportunities	**Threats**
• May activate the applicants that do not select code tests as their participation route to make improvements to their application letters. • Engages the applicants in a whole new way. • Allows applicants with less or none proof of evidence to have a chance of receiving a participation ticket to the event. • Can be used to eliminate the candidates that are less interested in the event. A more interested candidate is willing to invest more time in applying. • Great for marketing. • Offers a new way of participation.	• The right answers may leak to audiences before applying deadline has passed, affecting the validity of the results. • May affect the brand image if poorly made. • Candidates may feel like it is too much work to do a code test. Can be also seen as an opportunity, as only motivated candidates apply. • Candidates unwilling to do a code test may not apply because they expect the code test participation method to be preferred over applying with documents.

Opportunities and Threats (O-T)

The narrowing of the applying methods leads to a situation where the analysis' external parts (O-T) should be considered, also listed in table 1. The use of code tests engages the applicants in a way that other applying methods do not reach. For an applicant it is most likely more interesting to screen their level of knowledge by taking a test, rather than to send some documents. Also, the applicants that were to send documentation might decide to improve their application letters to stand out from the other applicants competing for being hired. Another opportunity that the code tests offer for applicants is that even with lesser proof of evidence they are not overlooked and also have a chance of becoming employed.

When code tests are applied to events, there is likely an increase to the marketing potential. As previously stated, there was a substantial increase in the pool of candidates, which, deducted from comparing participant amounts of Coding Factory events to each other and the interview results, displayed in appendix 2, is a direct cause of the addition of code tests.

There are some downsides to using code tests in events as well. Answering to a code test may also be considered arduous by some applicants. Some of them might not apply, because they would expect this participation method to be preferable and be put off from doing a test.

Also, if only the tests were inspected, and purely based on those results the participants were chosen some applicants may work in groups and their cooperation with one another would affect the validity of the results, as the test was aimed to grade a single candidate's performance. Another threat is that the right answers may leak to third parties, which would vitiate the code tests altogether. This would directly affect the organizer company's brand image heavily. Another concern that would deteriorate the brand image is if the questions were so poorly made that company would lose recognition in the IT industry.

5 Discussion

As code tests are a newly introduced method of evaluation in IT recruitment process, which was mentioned earlier in section 2.3.2, there does not exist a lot of prior research that the results of this study could be compared with. It is difficult to determine whether the made decisions, regarding the methods and material, were suitable for mapping the usefulness of code tests or not. However, as the goal was to find out if code tests are a useful part of an event's candidate evaluation and how the questions themselves should be made, these the chosen approaches did manage to reveal with reasonable reliability.

The reason for questioning the validity of the results is that the project was carried with haste due to its comprehensive scope and the limited amount of time available. These factors determined the materials and methods used, and as an afterthought I would choose a different approach altogether to one that would likely yield more trustworthy results. The qualitative portion of the material, the interview, proved its worth and I would probably try contacting a few more participants for a phone interview. As there are a lot of other ways of gathering qualitative data too, I would not limit this research to just interviews but, instead, structure two simple forms. One of these would be sent to all event participants and the other to the job applicants in the registry of aTalent Recruiting. The aim of these forms would be to compare how people who have taken a code test perceive these tests differently to those who have not.

From this data the deductions of the usability of code tests in general would be made. The quality of questions would be measured as in the execution strategy used, but with lesser weight on the analysis of the Coding Factory 2017 event's code test questions. The reasoning for this is that the information from these questions did not contribute to this research as there were restrictions set by the client company. Additionally, narrowing the project's goals down to a more reasonable level would ease in delving into code tests more.

Theoretically, the more material on hand, the less erroneous conclusions can be made. As mentioned, the amount of respondents that were interviewed was only one, and this number should be considerably higher to make the interview's results more dependable. Also, the research material was gathered from sources that do not have relation to one another. As an example of an execution strategy that might work better than the one used is comparing applicants' existing portfolios, such as GitHub and Stack Overflow profiles, to that person's performance in a code test. This alternative approach might considerably increase the reliability of one sector of the research, the reliability of code test results, but as it takes time to do more in-depth comparisons, this option had to be bypassed. As described in section 2.2.3, existing data is the most trustworthy way to evaluate applicant's level of skill. However, as this research tried to achieve more than just approximate the usability of code tests, a compromise had to be made. This means that the chosen strategy was not selected because it would yield the most accurate data but because it is simply the quickest one to get any data.

It was mentioned before in section 2.2 that recruitment industry is a competitive line of business and code tests are to ease technical assessment, enabling faster hiring. The research results would confirm these statements, meaning that code tests' ability to effectively set apart the most capable candidates from the rest is a valid argument. Referencing back to section 2.4.1, code tests also fairly accurately measure an applicant's adaptivity to do tasks. The results from the SWOT analysis and the CRESST criteria second this supposition, as there are a plentiful of different types of questions and adaptations that may be done to achieve variance in tests.

The decision of having a commonly applied method for question quality approximation was acceptable. Even though the CRESST criteria is a method that teachers use when creating their exams, it proved to be a valuable addition to the research. On the whole, the CRESST criteria filled its initial purpose sufficiently. It acted well in identifying whether the question was meaningful, language appropriate, fair and reliable, but it was difficult to come up with a solid division which cognitive complexity a question had. All and all, it was possible to identify the well formed questions from the ones that are not adequate as a part of a test, which was its objective. What is really significant in this realisation is that further development of code tests could gain influences from standards used by teachers when they author exams.

The material used in SWOT analysis was composed from a combination of research materials. The theoretical advantages and disadvantages of using assessment tests, the phone interview and Coding Factory events' data are the main resources of this substance. The analysis proved that while there are multiple downsides to code tests as well, they are, almost without exception, caused by poor contemplation and lack of knowledge regarding correct structure. What observation could be made from this is that code tests can also go horribly wrong and have serious consequences regarding the brand image. Therefore, only a capable programmer can structure a well functioning code tests, and that, only then, code tests should be considered as an option to use as a part of the recruitment process for programming job openings.

Another thing that the phone interview and SWOT analysis revealed was that, unlike initially expected from the theoretical research of code tests in section 2.4, code tests are not considered nearly as irritating or demanding a task when applied to events. The interviewed participant even stated that doing a code test is lesser of an assignment than writing an application letter. Overall, the use of code tests should be advised based on this data, rather than avoided. These recognitions would verify that the SWOT analysis did clarify the functionality of code tests and when to include them as a part of the evaluation process.

6 Conclusion

The first goal of the research was to ascertain the practical gains of code tests, both as a component of the evaluation process in default IT job openings, and as a part of the assessment procedures in events. Another goal was to disentangle the structure of well schemed test questions.

As a summary, the biggest achievement of this study was in finding and modeling the differences of individual questions and the conclusions driven from these deductions. These include, among others, that each question has to be selected regarding the job opening, the questions have to be uniform, appropriate and understandable, trick questions should not have a place in a test, the points a question yields should be with respect to the question's difficulty, and lastly, to uphold the interest of the applicant during the test, the questions have to have variance between them. These results could later be applied to the needs of the company in upcoming development of code tests for events and the tests used to evaluate the programming skills of an applicant for client companies.

As said in the previous section, I faced difficulties finding existing studies of the general use of code tests, which means that this thesis may be one of the first research papers that focuses in studying the right kind of structure for code tests. The results of the portion of the research that were to evaluate the general usability of code tests are extensively based on a limited sample size, and for that reason its trustworthiness is disputable. However, the aforementioned data is also backed by existing basic theoretical information and for that reason it can be considered reliable.

In similar future studies, the material utilised should be considerably broader and its lack of coherence should be addressed with cross-referenceable research material. By this I mean that an execution strategy similar to the one proposed in the previous section, where the applicants' portfolios are compared to code test results would likely yield more dependable information.

Bibliography

1 Fenton N. Software metrics : a rigorous and practical approach. Boca Raton, FL: CRC Press; 2014.

2 Howland J. Meaningful learning with technology. Boston: Pearson; 2012.

3 Statista. Internet of Things (IoT): number of connected devices worldwide from 2012 to 2020 (in billions); 2017. Available from: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ [cited February 02, 2017].

4 Fields M. Indispensable employees : how to hire them, how to keep them. Franklin Lakes, NJ: Career Press; 2001.

5 Avram A. IDC Study: How Many Software Developers Are Out There? InfoQ; 2014. Available from: https://www.infoq.com/news/2014/01/IDC-software-developers [cited February 07, 2017].

6 McCuller P. How to recruit and hire great software engineers : building a crack development team. Place of publication not identified New York: Apress Distributed to the Book trade worldwide by Springer; 2012.

7 Rouzer W. Mainframe Programmers Look to Avoid Fate of the Dinosaur. 1992;26:138–138.

8 Koncept Analytics. Global Recruitment Market Report: 2016 Edition - New Report by Koncept Analytics; 2016.

9 Broughton A. The use of social media in the recruitment process. Place of publication not identified, London: Acas; 2013.

10 Ciett. Ciett - Economic Report 2016 Edition. wecglobal.org; 2016. Available from: http://www.wecglobal.org/economicreport2016/ [cited February-01, 2017].

11 Knowlson I. Top Recruitment Sectors for Growth in 2016. Recruitment Buzz; 2016. Available from: http://recruitmentbuzz.co.uk/top-recruitment-sectors-for-growth-in-2016/ [cited March 03, 2017].

12 World Economy Forum. WEF - The Future of Jobs. World Economic Forum; 2016. Available from: http://www3.weforum.org/docs/WEF_Future_of_Jobs.pdf [cited February-09, 2017].

13 Elliot E. Why Hiring is So Hard in Tech. medium.com; 2015. Available from: https://medium.com/javascript-scene/why-hiring-is-so-hard-in-tech-c462c3230017#.ql2hinknr [cited February 09, 2017].

14 Viitala R. Henkilöstöjohtaminen. 4th ed. Edita; 2009.

15 Daie A. Tech Recruiting Best Practices. Khosla Ventures; 2015. Record of a seminar held for recruiters. Available from: https://www.youtube.com/watch?v=nzZuPI84jQI.

16 Markoff N. aTalent Recruiting; 2017. personal communication.

17 Laakso R. Rekrytointitutkimus 2014. aTalent Recruiting; 2014.

18 Räihä M. Evaluation of a successful recruitment process of senior managers in central government agencies of Finland : Case: Central government agencies of Finland [B.S. Thesis]. Haaga-Helia University of Applied Sciences. Ratapihantie 13, FI-00520 Helsinki, Finland; 2014.

19 Monster com. Standard Job Adds For Recruiting Employees. Monster.com;. Available from: http://hiring.monster.com/recruitment/Standard-Postings.aspx [cited February 21, 2017].

20 Rouse M. LinkedIn. WhatIs.com; 2016. Available from: http://whatis.techtarget.com/definition/LinkedIn [cited March 03, 2017].

21 Rouse M. GitHub. WhatIs.com; 2016. Available from: http://searchitoperations.techtarget.com/definition/GitHub [cited March 03, 2017].

22 Stack Exchange. About - Stack Exchange. Stack Exchange; 2017. Available from: http://stackexchange.com/about [cited March 03, 2017].

23 Odom CL. Hiring the Best Candidate Not the Best Resume. Financial Executive. 2013 03;29(2):61–63. Copyright - Copyright Financial Executives International Mar 2013; Last updated - 2017-02-18; SubjectsTermNotLitGenreText - United States–US. Available from: https://search-proquest-com.ezproxy.metropolia.fi/docview/1428387826?accountid=11363.

24 Cut-e. Online assessment built on robust psychometrics. Cut-e;. Available from: http://www.cut-e.com/online-assessment/ [cited February 18, 2017].

25 Camp R. Strategic interviewing : how to hire good people. San Francisco: Jossey-Bass; 2001.

26 Clay B. Is This a Trick Question? A Short Guide to Writing Effective Test Questions. Kansas: Kansas Curriculum Center; 2001.

27 Mansoor N. Enhancing thinking skills: Domain specific/Domain general strategies — A dilemma for science education. Instructional Science. 1994;22(6):413–422. Available from: http://www.jstor.org/stable/23370002.

28 Collins English Dictionary. Definition of ’programming skills’. Collins English Dictionary. Copyright © HarperCollins Publishers; 2017. Available from: https://www.collinsdictionary.com/dictionary/english/programming-skills [cited February 05, 2017].

29 Johnson P. 7 things every new programmer should know. ITworld; 2015. Available from: http://www.itworld.com/article/2969851/careers/7-things-every-new-programmer-should-know.html [cited February 14, 2017].

30 Torvalds L; 2006. private communication. Available from: https://lwn.net/Articles/193245/.

31 Nobari Sabina, Zadeh Davood. Designing a fuzzy model for decision support systems in the selection and recruitment process. 2013 apr;7:1486–1491.

32 RELEX. RELEX: System & Network Administrator. aTalent Recruiting; 2017. Available from: https://atalent.fi/avoimet-tyopaikat/system-network-administrator/ [cited February 05, 2017].

33 Winter T. 3 Ways to Use Coding Tests in Developer Recruitment. Tech.co; 2016. Available from: http://tech.co/ways-use-coding-tests-developer-recruitment-2016-01 [cited February 08, 2016].

34 Codility. Codility;. Available from: https://codility.com/pricing/ [cited February 09, 2017].

35 ProProfs. ProProfs;. Available from: http://www.proprofs.com/quiz-school/ [cited February 09, 2017].

36 Winter T. The 5 Essential Tactics to Recruiting Tech Talent. SmartRecruiters; 2016. Available from: https://www.smartrecruiters.com/blog/the-5-essential-tactics-to-recruiting-tech-talent/ [cited March 02, 2017].

37 Savage B. Why Coding Tests Are A Bad Interview Technique. BrandonSavage.net; 2009. Available from: https: //www.brandonsavage.net/why-coding-tests-are-a-bad-interview-technique/ [cited March 27, 2017].

38 White E. Interviewing Programmers. Eliś Ramblings; 2008. Available from: https://eliw.wordpress.com/2008/12/04/interviewing-programmers/ [cited March 27, 2017].

39 Woodcock B. How to pass graduate aptitude tests. University of Kent; 2015. Available from: https://www.kent.ac.uk/careers/psychotests.htm [cited March 02, 2017].

40 Woodcock B. Computer Programming Aptitude Test. University of Kent;. Available from: https://www.kent.ac.uk/careers/tests/computer-test.htm [cited March 02, 2017].

41 Psychometrics I T Tests. Berger Aptitude for Programming Test Advanced Form. Psychometrics I.T. Tests; 2009. Available from: http://www.psy-test.com/Baptaf.html [cited March 02, 2017].

42 Aldridge J. Benefits and Challenges of Timed Tests. Testshop; 2015. Available from: http://www.testshop.com/blog/benefits-and-challenges-of-timed-tests [cited March 23, 2017].

43 White C. Strategic management. New York: Palgrave Macmillan; 2004.

44 Codility. Codility Limited Terms of Service. Codility; 2016. Available from: https://codility.com/terms-of-service-for-companies/ [cited March 09, 2017].

1 Phone interview questions

Coding Factory 2017 candidate interview on the evaluation model

1. How did the entire candidate evaluation process feel like?
    a. Did you feel like the options given (resume + CV / code test) were sufficient and functional?
    b. Was the process humane, did the process and communication with the organizer feel distant? Would some other route for applying have been easier or functional (for example interview before the event)?
2. Regarding your current capabilities, did you feel that you could show the level of your knowledge and skills during the applying process?
3. Was the process too complex or long?
4. How would you describe your application experience in comparison to the ones you have had before?
5. Did you feel that the application process was more demanding that a regular one?
6. Before choosing how to apply, did you put thought into which applying method would yield the best chances of getting to participate in the event?

2 Phone interview answers

Coding Factory 2017 candidate interview on the evaluation model:

1. The applying process worked just fine, only problem being, that some of the questions were not unambiguous. The process was clear and easy. Based on the conversations I had had with the other participants I felt that all the candidates were well-suited and so the process was clearly functional.
    a. More visibility, otherwise the process was clear. I would remove the essay assignments or clarify them a lot. For example, one of the questions was ambiguous and it was not obvious how in-depth an answer was expected. There should be some restrictions for instance with word count or something. Also, it was a bit unclear how much time I was supposed to invest in this question. One last this, it was not possible to go to the previous question after proceeding. The communication worked just fine.
    b. I do not think any changes are warranted. If I am required to express my motivation, it is easier to do so in a written form.
2. Otherwise everything was ok, but some changes could be made to the questions asked. They were not necessary measuring the right things. Some were ok, but not all. Maybe the questions could have been less about nuances or quirks of the language and instead focus on programming in a more general manner.
3. It was not, I was only required to send a CV and do a test, the process was straightforward and functional.
4. The application process gave me the feeling that I had a better chance to prove my competence, regardless of the size of my pre-existing portfolio.
5. On the contrary, I felt that it was less burdening, as it is easier for me to answer questions, than to make guesses about what the recruiter is interested in knowing about me and trying to write something based on that into my application letter – which was something not required by this application process.
6. No, I did not. I chose the applying method which seemed to give me the highest chance of getting picked with the lowest amount of effort. However, it didn't seem like it made much of a difference for my chances which application method I chose, which made the choice easier. And I think that's how it should be.

3 General data from all Coding Factory 2017 tests

Java

Date	Score %	Score	Time Taken		Time Taken (hh:mm:ss)
01-16-2017 06:33:04 PM	80	16	26 min 50 sec	2:50:00	0:26:50
01-3-2017 05:48:48 PM	75	15	28 min 45 sec	4:45:00	0:28:45
01-17-2017 07:32:40 PM	70	14	27 min 29 sec	3:29:00	0:27:29
01-5-2017 01:28:40 PM	70	14	26 min 50 sec	2:50:00	0:26:50
12-20-2016 03:06:58 PM	70	14	28 min 9 sec	4:09:00	0:28:09
01-20-2017 09:50:49 PM	65	13	17 min 5 sec	17:05:00	0:17:05
01-11-2017 06:52:29 PM	65	13	23 min 56 sec	23:56:00	0:23:56
01-9-2017 03:39:58 PM	65	13	15 min 48 sec	15:48:00	0:15:48
01-17-2017 11:57:48 PM	60	12	29 min 31 sec	5:31:00	0:29:31
01-16-2017 02:35:48 PM	60	12	30 min	6:00:00	0:30:00
01-3-2017 04:06:50 PM	60	12	25 min 48 sec	1:48:00	0:25:48
12-16-2016 11:50:04 AM	60	12	22 min 48 sec	22:48:00	0:22:48
01-19-2017 05:58:09 PM	55	11	25 min 52 sec	1:52:00	0:25:52
01-18-2017 07:00:46 PM	55	11	29 min 52 sec	5:52:00	0:29:52
01-17-2017 07:52:05 PM	55	11	22 min 31 sec	22:31:00	0:22:31
01-14-2017 02:03:43 PM	55	11	30 min	6:00:00	0:30:00
01-11-17 12:32	55	11	22 min 38 sec	22:38:00	0:22:38
01-3-2017 07:11:23 PM	55	11	30 min	6:00:00	0:30:00
12-17-2016 05:35:02 AM	55	11	28 min 51 sec	4:51:00	0:28:51
01-19-2017 10:37:45 PM	50	10	29 min 49 sec	5:49:00	0:29:49
01-18-2017 10:23:02 PM	50	10	24 min 36 sec	0:36:00	0:24:36
01-17-2017 09:30:37 PM	50	10	30 min	6:00:00	0:30:00
01-10-2017 09:30:26 AM	50	10	26 min 11 sec	2:11:00	0:26:11
01-19-2017 06:10:46 PM	45	9	22 min 37 sec	22:37:00	0:22:37
01-17-2017 08:14:04 PM	45	9	30 min	6:00:00	0:30:00
12-29-2016 03:50:02 PM	45	9	18 min 29 sec	18:29:00	0:18:29
01-17-2017 11:58:33 PM	40	8	23 min 31 sec	23:31:00	0:23:31
01-17-2017 07:49:34 PM	40	8	30 min	6:00:00	0:30:00
01-03-17 11:23	40	8	22 min 24 sec	22:24:00	0:22:24
01-19-2017 06:35:25 PM	30	6	9 min 52 sec	9:52:00	0:09:52
01-18-2017 06:49:44 PM	30	6	16 min 44 sec	16:44:00	0:16:44
01-17-2017 12:21:33 PM	30	6	14 min 23 sec	14:23:00	0:14:23
01-16-2017 11:46:29 AM	30	6	10 min 5 sec	10:05:00	0:10:05
12-28-2016 03:37:42 PM	30	6	30 min	6:00:00	0:30:00
01-18-2017 07:52:28 PM	25	5	30 min	6:00:00	0:30:00
01-4-2017 06:20:48 PM	25	5	10 min 23 sec	10:23:00	0:10:23
	51.11111111	10.22222222			0:24:13
Amount				36	

Global

Date	Score %	Score	Time Taken		Time Taken (hh:mm:ss)
01-12-17 12:26	75	15	30 min	6:00:00	0:30:00
12-23-2016 02:53:54 PM	75	15	30 min	6:00:00	0:30:00
12-21-2016 12:32:07 PM	75	15	29 min 24 sec	5:24:00	0:29:24
01-17-2017 08:47:36 PM	70	14	30 min	6:00:00	0:30:00
01-16-2017 08:00:04 PM	70	14	29 min 35 sec	5:35:00	0:29:35
01-27-2017 10:09:01 AM	65	13	30 min	6:00:00	0:30:00
01-4-2017 03:23:17 PM	65	13	28 min 8 sec	4:08:00	0:28:08
01-21-2017 05:57:36 AM	60	12	29 min 15 sec	5:15:00	0:29:15
01-20-2017 11:39:18 AM	60	12	27 min 23 sec	3:23:00	0:27:23
01-17-2017 06:49:32 PM	60	12	28 min 8 sec	4:08:00	0:28:08
01-4-2017 05:53:18 PM	60	12	30 min	6:00:00	0:30:00
12-29-2016 02:41:27 PM	60	12	30 min	6:00:00	0:30:00
01-20-2017 11:42:50 AM	55	11	26 min 20 sec	2:20:00	0:26:20
01-17-2017 08:05:39 PM	55	11	29 min 54 sec	5:54:00	0:29:54
01-17-2017 06:33:49 PM	55	11	29 min 49 sec	5:49:00	0:29:49
01-6-2017 09:07:08 AM	55	11	29 min 55 sec	5:55:00	0:29:55
12-22-2016 06:18:38 PM	55	11	30 min	6:00:00	0:30:00
12-17-2016 09:04:42 PM	55	11	19 min 54 sec	19:54:00	0:19:54
12-15-2016 12:36:17 PM	55	11	30 min 17 sec	6:17:00	0:30:17
01-19-2017 10:37:57 PM	50	10	24 min 56 sec	0:56:00	0:24:56
01-19-2017 01:36:07 PM	50	10	29 min 52 sec	5:52:00	0:29:52
01-18-2017 05:51:35 PM	50	10	29 min15 sec	5:15:00	0:29:15
01-13-2017 06:29:53 PM	50	10	30 min	6:00:00	0:30:00
01-13-2017 12:03:13 AM	50	10	29 min 14 sec	5:14:00	0:29:14
12-17-2016 09:50:59 AM	50	10	30 min	6:00:00	0:30:00
01-19-2017 07:51:21 PM	45	9	30 min	6:00:00	0:30:00
01-19-2017 07:42:32 PM	45	9	30 min	6:00:00	0:30:00
01-18-2017 10:03:36 PM	45	9	30 min	6:00:00	0:30:00
01-16-2017 09:06:44 AM	45	9	30 min	6:00:00	0:30:00
12-20-2016 03:26:30 PM	45	9	17 min 26 sec	17:26:00	0:17:26
12-15-2016 09:17:49 PM	45	9	30 min	6:00:00	0:30:00
01-18-2017 10:45:31 PM	40	8	28 min 35 sec	4:35:00	0:28:35
01-17-2017 11:34:31 PM	40	8	27 min 49 sec	3:49:00	0:27:49
01-19-2017 10:48:01 AM	35	7	25 min 5 sec	1:05:00	0:25:05
01-17-2017 11:44:53 PM	35	7	18 min 17 sec	18:17:00	0:18:17
01-04-17 10:29	35	7	30 min	6:00:00	0:30:00
01-20-2017 11:46:32 AM	30	6	30 min	6:00:00	0:30:00
01-19-2017 04:45:06 PM	30	6	16 min 3 sec	16:03:00	0:16:03
01-17-2017 06:22:29 PM	30	6	30 min	6:00:00	0:30:00
12-23-2016 11:45:37 AM	30	6	30 min	6:00:00	0:30:00
01-19-2017 10:22:55 PM	25	5	27 min 32 sec	3:32:00	0:27:32
01-19-2017 10:53:41 AM	25	5	30 min	6:00:00	0:30:00
01-19-2017 10:40:50 PM	20	4	29 min 27 sec	5:27:00	0:29:27
01-19-2017 10:25:32 PM	20	4	30 min	6:00:00	0:30:00
	48.75	9.75			0:28:13
Amount			44		

JavaScript

Date	Score %	Score	Time Taken		Time Taken (hh:mm:ss)
01-5-2017 04:30:23 PM	90	18	25 min 49 sec	1:49:00	0:25:49
12-20-2016 09:19:34 PM	80	16	25 min 7 sec	1:07:00	0:25:07
01-4-2017 01:08:26 PM	75	15	22 min 17 sec	22:17:00	0:22:17
12-23-2016 03:31:09 PM	75	15	29 min 50 sec	5:50:00	0:29:50
01-19-2017 04:36:00 PM	65	13	30 min	6:00:00	0:30:00
01-19-2017 12:29:28 PM	65	13	27 min 58 sec	3:58:00	0:27:58
01-16-2017 05:10:36 PM	65	13	27 min 13 sec	3:13:00	0:27:13
01-17-2017 04:09:22 PM	60	12	16 min 39 sec	16:39:00	0:16:39
01-2-2017 04:14:01 PM	60	12	30 min	6:00:00	0:30:00
12-29-2016 11:06:30 AM	60	12	25 min 57 sec	1:57:00	0:25:57
12-19-2016 10:16:51 PM	60	12	30 min	6:00:00	0:30:00
01-17-2017 11:45:49 PM	55	11	30 min	6:00:00	0:30:00
01-6-2017 04:08:03 PM	55	11	27 min 16 sec	3:16:00	0:27:16
01-18-2017 06:59:34 PM	50	10	29 min 59 sec	5:59:00	0:29:59
01-18-2017 04:30:01 PM	50	10	27 min 10 sec	3:10:00	0:27:10
01-18-2017 03:37:34 AM	50	10	29 min 46 sec	5:46:00	0:29:46
12-15-2016 06:01:55 PM	50	10	30 min	6:00:00	0:30:00
01-19-2017 06:21:25 PM	45	9	18 min 41 sec	18:41:00	0:18:41
12-27-2016 03:30:31 PM	45	9	30 min	6:00:00	0:30:00
01-02-17 22:51	40	8	30 min	6:00:00	0:30:00
12-17-2016 11:06:54 PM	40	8	27 min14 sec	3:14:00	0:27:14
12-16-2016 04:05:31 PM	40	8	30 min	6:00:00	0:30:00
01-18-2017 07:36:22 PM	35	7	26 min 55 sec	2:55:00	0:26:55
01-11-17 10:59	35	7	30 min	6:00:00	0:30:00
01-19-2017 10:31:47 PM	30	6	30 min	6:00:00	0:30:00
01-18-2017 07:27:34 PM	30	6	22 min 8 sec	22:08:00	0:22:08
01-11-17 11:52	30	6	29 min 22 sec	5:22:00	0:29:22
12-23-2016 01:36:21 AM	30	6	30 min	6:00:00	0:30:00
01-18-2017 07:07:17 PM	15	3	10 min15 sec	10:15:00	0:10:15
	51.03448276	**10.20689655**			**0:26:53**
Amount			29		

Prosess length

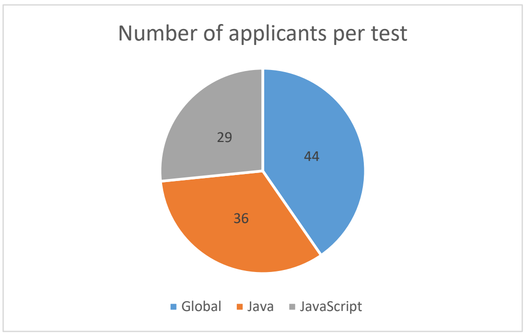
Request se	Test send c	Test finish date	Total (Days	Req - Test (	Test - Finish (Days)
15-12-16	15-12-16	20.12.2016	5	0	5
15-12-16	19-12-16	22.12.2016	7	4	3
15-12-16	15-12-16	29.12.2016	14	0	14
15-12-16	15-12-16	21.12.2016	6	0	6
15-12-16					
15-12-16					
15-12-16	19-12-16	27.12.2016	12	4	8
15-12-16	15-12-16	16.12.2016	1	0	1
15-12-16	15-12-16	17.12.2016	2	0	2
15-12-16	15-12-16	16.12.2016	1	0	1
15-12-16	15-12-16			0	
15-12-16	15-12-16			0	
15-12-16	15-12-16			0	
15-12-16	15-12-16			0	
15-12-16	15-12-16	16.12.2016	1	0	1
15-12-16	16-12-16	17.12.2016	2	1	1
15-12-16	15-12-16	17.12.2016	2	0	2
15-12-16					
15-12-16					
15-12-16	15-12-16	20.12.2016	5	0	5
15-12-16	15-12-16			0	
15-12-16	16-12-16	2.1.2017	18	1	17
15-12-16	15-12-16	29.12.2016	14	0	14
19-12-16	22-12-16	22.12.2016	3	3	0
19-12-16	19-12-16	19.12.2016	0	0	0
19-12-16	19-12-16	28.12.2016	9	0	9
19-12-16	22-12-16			3	
19-12-16	20-12-16	20.12.2016	1	1	0
20-12-16					
20-12-16	04-01-17	4.1.2017	15	15	0
20-12-16	22-12-16	6.1.2017	17	2	15
20-12-16	22-12-16	23.12.2016	3	2	1
20-12-16	22-12-16	23.12.2016	3	2	1
22-12-16	04-01-17	5.1.2017	14	13	1
22-12-16	22-12-16	23.12.2016	1	0	1
22-12-16	04-01-17	12.1.2017	21	13	8
27-12-16	02-01-17	3.1.2017	7	6	1
27-12-16	28-12-16	29.12.2016	2	1	1
27-12-16	28-12-16			1	
27-12-16	28-12-16	2.1.2017	6	1	5
29-12-16	02-01-17	3.1.2017	5	4	1
02-01-17	03-01-17			1	
02-01-17	03-01-17	3.1.2017	1	1	0
02-01-17	04-01-17	13.1.2017	11	2	9
02-01-17	16-01-17	18.1.2017	16	14	2
03-01-17	04-01-17	5.1.2017	2	1	1
03-01-17	04-01-17	4.1.2017	1	1	0
03-01-17	04-01-17	4.1.2017	1	1	0
03-01-17	04-01-17			1	
03-01-17	04-01-17	4.1.2017	1	1	0
04-01-17	04-01-17	6.1.2017	2	0	2
04-01-17	04-01-17	4.1.2017	0	0	0
09-01-17	10-01-17	11.1.2017	2	1	1
09-01-17	09-01-17	17.1.2017	8	0	8
09-01-17	09-01-17	11.1.2017	2	0	2
09-01-17	09-01-17			0	

09-01-17	09-01-17	11.1.2017	2	0	2
09-01-17					
09-01-17	09-01-17	16.1.2017	7	0	7
09-01-17	09-01-17	17.1.2017	8	0	8
09-01-17	09-01-17			0	
09-01-17	10-01-17	17.1.2017	8	1	7
09-01-17	09-01-17	10.1.2017	1	0	1
09-01-17	09-01-17	17.1.2017	8	0	8
09-01-17	09-01-17	10.1.2017	1	0	1
09-01-17	10-01-17	17.1.2017	8	1	7
09-01-17	09-01-17	16.1.2017	7	0	7
09-01-17	10-01-17	11.1.2017	2	1	1
11-01-17	12-01-17	14.1.2017	3	1	2
11-01-17	12-01-17			1	
11-01-17	13-01-17	13.1.2017	2	2	0
11-01-17	12-01-17	17.1.2017	6	1	5
12-01-17	16-01-17	17.1.2017	5	4	1
12-01-17					
12-01-17					
13-01-17	16-01-17			3	
13-01-17	16-01-17	17.1.2017	4	3	1
13-01-17	16-01-17	16.1.2017	3	3	0
13-01-17	16-01-17	17.1.2017	4	3	1
13-01-17	13-01-17	16.1.2017	3	0	3
13-01-17	16-01-17	16.1.2017	3	3	0
13-01-17	16-01-17			3	
16-01-17	16-01-17	16.1.2017	0	0	0
16-01-17	16-01-17	17.1.2017	1	0	1
16-01-17	16-01-17			0	
16-01-17	16-01-17	17.1.2017	1	0	1
16-01-17	16-01-17	17.1.2017	1	0	1
16-01-17	16-01-17	18.1.2017	2	0	2
16-01-17	20-01-17	20.1.2017	4	4	0
16-01-17	16-01-17			0	
16-01-17	16-01-17	16.1.2017	0	0	0
16-01-17	16-01-17			0	
16-01-17	17-01-17	17.1.2017	1	1	0
17-01-17	19-01-17			2	
17-01-17	19-01-17			2	
17-01-17					
17-01-17					
17-01-17	18-01-17			1	
17-01-17	17-01-17	17.1.2017	0	0	0
17-01-17	17-01-17	17.1.2017	0	0	0
17-01-17	18-01-17	19.1.2017	2	1	1
18-01-17	18-01-17	18.1.2017	0	0	0
18-01-17	18-01-17	18.1.2017	0	0	0
18-01-17	18-01-17	18.1.2017	0	0	0
18-01-17	18-01-17	18.1.2017	0	0	0
18-01-17					
18-01-17	18-01-17	18.1.2017	0	0	0
18-01-17	19-01-17	19.1.2017	1	1	0
18-01-17	18-01-17	18.1.2017	0	0	0
18-01-17	18-01-17			0	
18-01-17	18-01-17	18.1.2017	0	0	0
18-01-17	18-01-17	18.1.2017	0	0	0
18-01-17					
18-01-17	19-01-17	19.1.2017	1	1	0
18-01-17	19-01-17			1	

18-01-17	18-01-17	18.1.2017	0	0	0
18-01-17	18-01-17	18.1.2017	0	0	0
18-01-17					
18-01-17	18-01-17	18.1.2017	0	0	0
18-01-17	18-01-17			0	
18-01-17					
19-01-17					
19-01-17					
19-01-17	19-01-17	19.1.2017	0	0	0
19-01-17					
19-01-17					
19-01-17	19-01-17			0	
19-01-17	19-01-17	19.1.2017	0	0	0
19-01-17					
19-01-17					
19-01-17	19-01-17	19.1.2017	0	0	0
19-01-17	19-01-17	19.1.2017	0	0	0
19-01-17	19-01-17			0	
19-01-17	19-01-17			0	
19-01-17	19-01-17	20.1.2017	1	0	1
19-01-17	19-01-17	19.1.2017	0	0	0
19-01-17	19-01-17	19.1.2017	0	0	0
19-01-17	19-01-17			0	
19-01-17	19-01-17			0	
19-01-17	19-01-17			0	
19-01-17	19-01-17			0	
19-01-17	19-01-17			0	
19-01-17	19-01-17	19.1.2017	0	0	0
19-01-17					
19-01-17	19-01-17			0	
19-01-17	19-01-17	20.1.2017	1	0	1
19-01-17	19-01-17			0	
19-01-17	19-01-17			0	
19-01-17	19-01-17	19.1.2017	0	0	0
19-01-17					
19-01-17	19-01-17	19.1.2017	0	0	0
19-01-17	19-01-17	19.1.2017	0	0	0
19-01-17	20-01-17			1	
19-01-17	20-01-17			1	
			3.59375	1.113636	2.28125

All applicants

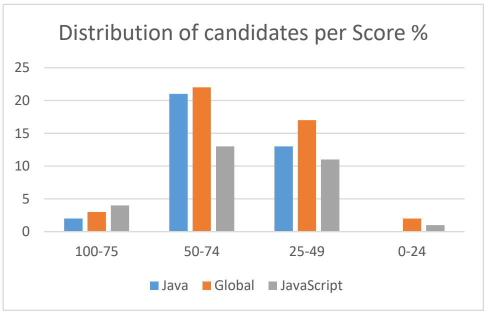
Type	Amount	Avg.Score '	Avg. Score (X/20)
All	109	50.29853	10.05970626
Global	44	48.75	9.75
Java	36	51.11111	10.22222222
JavaScript	29	51.03448	10.20689655



Summary

Code test drop-out %	Total Acc. %	Code test /	Non- Code test Acc. %
57.79816514	19.01140684	54	46

	Java	Global	JavaScript
100-75	2	3	4
50-74	21	22	13
25-49	13	17	11
0-24	0	2	1



Accepted applicants

Type	Amount	Score %	Score (X/20)
All	27	68.7037	13.74074074
Global	8	61.875	12.375
Java	12	65	13
JavaScript	7	72.14286	14.42857143

Total applicants to event

Type	
All	263
Code test	109
Total accepted	50
Code test accepted	27
% of Acc. With Code Test	24.7706422

4 Questions gathered from code test platforms

Example Questions:

JavaScript

1.

Provider: TestDome

Fix the bug.

```
function average(a, b) {
  return a + b / 2;
}
console.log(average(2, 1));
```

2.

Provider: Tests4Geeks

Which of the following JavaScript snippets will create an alert box containing "Hello John Doe"?

1.

```
var Person = function (firstName, lastName, age) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.age = age;
};

person = new Person("John", "Doe", "50");

Person.prototype.sayHello = function () {
  alert("Hello " + this.firstName + " " + this.lastName);
};

person.sayHello();
```

2.

```
var Person = function (firstName, lastName, age) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.age = age;
};

person = new Person("John", "Doe", "50");

Person.sayHello = function () {
  alert("Hello " + this.firstName + " " + this.lastName);
};

person.sayHello();
```

3.

```
var person = {
  firstName: “John”,
  lastName: “Doe”,
  age: “50”
};

person.sayHello = function () {
  alert("Hello " + this.firstName + " " + this.lastName);
};

person.sayHello();
```

4.

```
var person = {
  firstName: “John”,
  lastName: “Doe”,
  age: “50”
};

person.prototype.sayHello = function () {
  alert("Hello " + this.firstName + " " + this.lastName);
};

person.sayHello();
```

3.

Provider: ProgrammingSkills

Which popup box you use when want some information comes from user?

1.

`alert("Write some text here")`

2.

`confirm("Write some text here")`

3.

`prompt("Write some text here", "Write default value here")`

4.

Provider: ProgrammingSkills

How do you fetch the first span on the page, which has the class "green"?

1.

`$('span, .green, :first')`

2.

`$('first .green span')`

3.

`$('span.green:first')`

4.

`null`

5.

Provider: ProgrammingSkills

Which of the following jQuery method checks if event.stopPropagation() was ever called on this event object?

1.

`isDefaultPrevented()`

2.

`isPropagationStopped()`

3.

`isImmidatePropagationStopped()`

4.

`None of the above.`

6.

Provider: ProgrammingSkills

What is the result of:

```
<script language="javascript">
  function x() {
    document.write(2+5+"8");
  }
</script>
```

1.

`258`

2.

`Error`

3.

`7`

4.

`78`

7.

Provider: ProgrammingSkills

Look at the following selector: $("div"). What does it select?

1.

`The first div element`

2.

`All div elements`

8.

Provider: ProgrammingSkills

______method evaluates a string of JavaScript code in the context of the specified object.

1.

`Eval`

2.

`ParseInt`

3.

`ParseFloat`

4.

`Efloat`

9.

Provider: ProgrammingSkills

Which of the following is a JavaScript datatype?

1.

`Null`

2.

`Object`

3.

`Undefined`

4.

`All of the above`

10.

Provider: ProgrammingSkills

Which of the following function of Array object returns the last (greatest) index of an element within the array equal to the specified value, or -1 if none is found?

1.

`indexOf()`

2.

`lastIndexOf()`

3.

`join()`

4.

`Map()`

11.

Provider: ProgrammingSkills

What is the correct syntax for Array Declaration?

1.

`var city = new Array(“delhi”, “agra”, “akot”, “aligarh”);`

2.

`var city = [“delhi”, “agra”, “akot”, “Aligarh”];`

3.

`Both 1. and 2.`

4.

`None of them`

12.

Provider: ProgrammingSkills

JQuery is the programming language?

1.

`Yes`

2.

`No`

13.

Provider: ProgrammingSkills

Which of the following are the ways to create a Javascript Object?

1.

`Var obj = {};`

2.

`Var obj = new Object();`

3.

`Var obj = Object.create();`

4.

`All of the above`

14.

Provider: ProgrammingSkills

What is the use of "this" keyword in javascript?

1.

`It refers to current object`

2.

`It refers to previous object`

3.

`Is is a variable which contains value`

4.

`None of the above`

14.

Provider: ProgrammingSkills

Which of the following is a server-side JavaScript object?

1.

`FileUpLoad`

2.

`Function`

3.

`File`

4.

`Date`

12.

Provider: TestDome

Function appendChildren should add a new child div to each existing div. New divs should be decorated by calling decorateDiv.

For example, after appendChildren is executed, the following divs:

```
  <div id="a">
    <div id="b">
    </div>
  </div>
```

should take the following form (assuming decorateDiv does nothing):

```
  <div id="a">
    <div id="b">
      <div></div>
    </div>
    <div></div>
  </div>
```

The code below should do the job, but for some reason it goes into an infinite loop. Fix the bugs.

13.

Provider: TestDome

Fix the bugs in the registerHandlers function. An alert should display anchor's zero-based index within a document instead of following the link.

For example, in the document below, the alert should display "2" when Google anchor is clicked since it is the third anchor element in the document and its zero-based index is 2.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Test page</title>
  </head>
  <body>
```

```
    In my life, I used the following web search engines:<br/>

    <a href="//www.yahoo.com">Yahoo!</a><br/>

    <a href="//www.altavista.com">AltaVista</a><br/>

    <a href="//www.google.com">Google</a><br/>

  </body>

</html>
```

14.

Provider: TestDome

```
Implement the removeProperty function which takes an object and property name, and does the
following:


If the object obj has a property prop, the function removes the property from the object and
returns true; in all other cases it returns false.
```

15.

Provider: TestDome

```
The createProductCodeForm function is used to create a new form that accepts a product code from
a user.


The current version of the form contains the hint: 'The product code can be found on the label'.
This hint is currently always visible to the user.


Improve the form so that the hint is only rendered when the input element is the focused element.
```

Java

1.

Provider: IndiaBix

What is the value of "d" after this line of code has been executed?

```
Double d = Math.round(2.5 + Math.random());
```

1.

`2`

2.

`3`

3.

`4`

4.

`2.5`

2.

Provider: IndiaBix

What will be the output of the program?

```
public class CommandArgs {
  public static void main(String [] args) {
    String s1 = args[1];
    String s2 = args[2];
    String s3 = args[3];
    String s4 = args[4];
    System.out.print(" args[2] = " + s2);
  }
}
```

and the command-line invocation is

```
> java CommandArgs 1 2 3 4
```

1.

`Args[2] = 2`

2.

`Args[2] = 3`

3.

`Args[2] = null`

4.

`An exception is thrown at runtime.`

3.

Provider: IndiaBix

What will be the output of the program?

```
public class CommandArgsTwo {
  public static void main(String [] argh){
    int x;
    x = argh.length;
    for (int y = 1; y <= x; y++) {
      System.out.print(" " + argh[y]);
    }
  }
}
```

and the command-line invocation is

```
> java CommandArgsTwo 1 2 3
```

1.

`0 1 2`

2.

`1 2 3`

3.

`0 0 0`

4.

`An exception is thrown at runtime.`

4.

Provider: IndiaBix

Which statement is true for the class java.util.ArrayList?

1.

`The elements in the collection are ordered.`

2.

`The collection is guaranteed to be immutable.`

3.

`The elements in the collection are guaranteed to be unique.`

4.

`The elements in the collection are accessed using an unique key.`

5.

Provider: IndiaBix

What will be the output of the program?

```
public class X {
  public static void main(String [] args){
    try{
       badMethod(); /* Line 7 */
       System.out.print("A");
    } catch (Exception ex) /* Line 10 */{
       System.out.print("B"); /* Line 12 */
    } finally /* Line 14 */ {
       System.out.print("C"); /* Line 16 */
    }
  System.out.print("D"); /* Line 18 */
  }
  public static void badMethod(){
     throw new RuntimeException();
  }
}
```

1.

AB

2.

BC

3.

ABC

4.

BCD

6.

Provider: IndiaBix

What is the widest valid returnType for methodA in line 3?

```
public class ReturnIt{
  returnType methodA(byte x, double y) /* Line 3 */{
    return (long)x / y * 2;
  }
}
```

1.

int

2.

byte

3.

long

4.

double

7.

Provider: IndiaBix

What will be the output of the program?

```
class Base
```

```
{
    Base()
    {
        System.out.print("Base");
    }
}
public class Alpha extends Base
{
    public static void main(String[] args)
    {
        new Alpha(); /* Line 12 */
        new Base(); /* Line 13 */
    }
}
```

1.

`int`

2.

`byte`

3.

`long`

4.

`double`

8.

Provider: TestDome

Write a function that, given a list and a target sum, returns zero-based indices of any two distinct elements whose sum is equal to the target sum. If there are no such elements, the function should return null.

For example, findTwoSum(new int[] { 1, 3, 5, 7, 9 }, 12) should return a single dimensional array with two elements and contain any of the following pairs of indices:

1 and 4 (3 + 9 = 12)
2 and 3 (5 + 7 = 12)
3 and 2 (7 + 5 = 12)
4 and 1 (9 + 3 = 12)

9.
Provider: TestDome

Implement a function folderNames, which accepts a string containing an XML file that specifies folder structure and returns all folder names that start with startingLetter. The XML format is given in the example below.

For example, for the letter 'u' and an XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<folder name="c">
  <folder name="program files">
    <folder name="uninstall information" />
  </folder>
  <folder name="users" />
</folder>
```

the function should return a collection with items "uninstall information" and "users" (in any order).

10.
Provider: TestDome

Implement function countNumbers that accepts a sorted array of integers and counts the number of array elements that are less than the parameter lessThan.

For example, SortedSearch.countNumbers(new int[] { 1, 3, 5, 7 }, 4) should return 2 because there are two array elements less than 4.

11.
Provider: TestDome

Write a function that provides change directory (cd) function for an abstract file system.

Notes:

Root path is '/'.
Path separator is '/'.
Parent directory is addressable as "..".
Directory names consist only of English alphabet letters (A-Z and a-z).
For example, new Path("/a/b/c/d").cd("../x").getPath() should return "/a/b/c/x".

Note: Do not use built-in path-related functions.

12.
Provider: TestDome

Binary search tree (BST) is a binary tree where the value of each node is larger or equal to the values in all the nodes in that node's left subtree and is smaller than the values in all the nodes in that node's right subtree.

Write a function that checks if a given binary search tree contains a given value.

For example, for the following tree:

n1 (Value: 1, Left: null, Right: null)
n2 (Value: 2, Left: n1, Right: n3)
n3 (Value: 3, Left: null, Right: null)
Call to contains(n2, 3) should return true since a tree with root at n2 contains number 3.

13.
Provider: TestDome

A palindrome is a word that reads the same backward or forward.

Write a function that checks if a given word is a palindrome. Character case should be ignored.

For example, isPalindrome("Deleveled") should return true as character case should be ignored, resulting in "deleveled", which is a palindrome since it reads the same backward and forward.

14.
Provider: TestDome

A TrainComposition is built by attaching and detaching wagons from the left and the right sides.

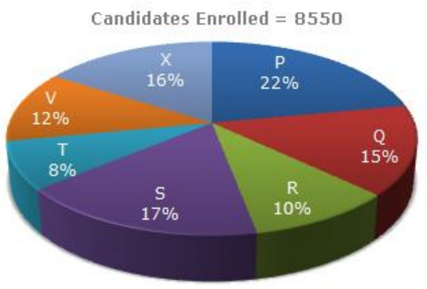
For example, if we start by attaching wagon 7 from the left followed by attaching wagon 13, again from the left, we get a composition of two wagons (13 and 7 from left to right). Now the first wagon that can be detached from the right is 7 and the first that can be detached from the left is 13.

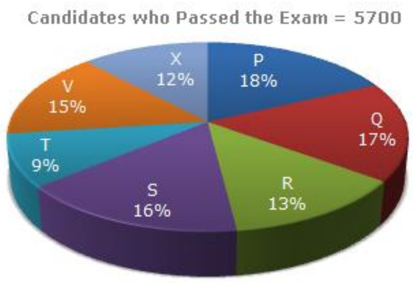
Implement a TrainComposition that models this problem.

DataBases

1.
Provider: IndiaBix
Study the following graph carefully and answer the questions given below:

Distribution of candidates who were enrolled for MBA entrance exam and the candidates (out of those enrolled) who passed the exam in different institutes:





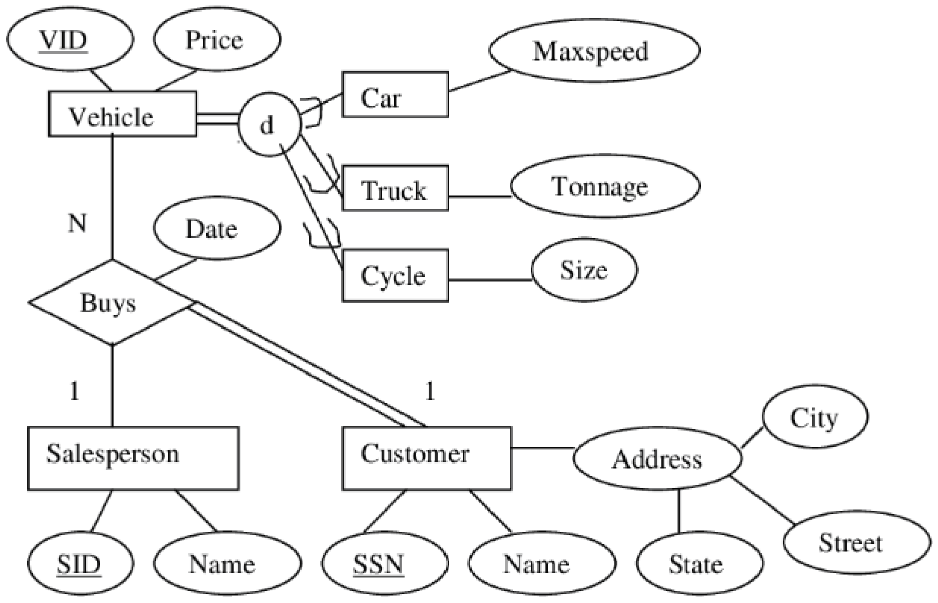
1. What percentage of candidates passed the Exam from institute T out of the total number of candidates enrolled from the same institute?
2. Which institute has the highest percentage of candidates passed to the candidates enrolled?
3. The number of candidates passed from institutes S and P together exceeds the number of candidates enrolled from institutes T and R together by:
4. What is the percentage of candidates passed to the candidates enrolled for institutes Q and R together?
5. What is the ratio of candidates passed to the candidates enrolled from institute P?

2.
Provider: TestDome
Select all queries that return number of students whose first name is John.

```
TABLE student
  id INTEGER NOT NULL PRIMARY KEY
  firstName VARCHAR(30) NOT NULL
```

```
lastName VARCHAR(30) NOT NULL
```

3.
Provider: docsity
Consider the following ER diagram for the next two questions.



1. Which of the following statements is true?
   a. A Customer does not have to participate in the Buys relationship
   b. One Vehicle can be considered both a Car and a Truck
   c. A Customer buys a vehicle from one Salesperson
   d. A Customer is identified by the combination of Street, City and State
2. Which of the following statements is true?
   a. A Customer can only buy vehicles from one salesperson
   b. A Salesperson can sell vehicles to only one Customer
   c. A given vehicle is sold by one Salesperson to one Customer
   d. All the above

5 Parsed code test questions

Questions organized according to their cognitive complexity:

Knowledge:

- Which popup box you use when want some information comes from user?
- How do you fetch the first span on the page, which has the class "green"?
- Which of the following jQuery method checks if event.stopPropagation() was ever called on this event object?
- ______method evaluates a string of JavaScript code in the context of the specified object.
- Which of the following is a JavaScript datatype?
- Which of the following function of Array object returns the last (greatest) index of an element within the array equal to the specified value, or -1 if none is found?
- What is the correct syntax for Array Declaration?
- JQuery is the programming language?
- What is the use of "this" keyword in javascript?
- Which of the following is a server-side JavaScript object?
- Which statement is true for the class java.util.ArrayList?
- Which of the following are the ways to create a Javascript Object?

Comprehension:

- Look at the following selector: $("div"). What does it select?
- Which of the following JavaScript snippets will create an alert box containing "Hello John Doe"?
- What is the result of:
  ```
  <script language="javascript">

    function x() {

      document.write(2+5+"8");

    }

  </script>
  ```
- What is the value of "d" after this line of code has been executed?
  ```
  Double d = Math.round(2.5 + Math.random());
  ```
- What will be the output of the program?
  ```
  public class CommandArgs {
    public static void main(String [] args) {
      String s1 = args[1];
      String s2 = args[2];
      String s3 = args[3];
      String s4 = args[4];
      System.out.print(" args[2] = " + s2);
    }
  }
  ```
  and the command-line invocation is
  ```
  > java CommandArgs 1 2 3 4
  ```
- What will be the output of the program?
  ```
  public class CommandArgsTwo {
    public static void main(String [] argh){
      int x;
      x = argh.length;
      for (int y = 1; y <= x; y++) {
        System.out.print(" " + argh[y]);
      }
    }
  }
  ```
  and the command-line invocation is

```
> java CommandArgsTwo 1 2 3
```

- What will be the output of the program?
```
public class X {
  public static void main(String [] args){
    try{
       badMethod(); /* Line 7 */
       System.out.print("A");
    } catch (Exception ex) /* Line 10 */{
       System.out.print("B"); /* Line 12 */
    } finally /* Line 14 */ {
       System.out.print("C"); /* Line 16 */
    }
  System.out.print("D"); /* Line 18 */
  }
  public static void badMethod(){
     throw new RuntimeException();
  }
}
```
- What is the widest valid returnType for methodA in line 3?
```
public class ReturnIt{
  returnType methodA(byte x, double y) /* Line 3 */{
    return (long)x / y * 2;
  }
}
```
- What will be the output of the program?
```
class Base
{
    Base()
    {
        System.out.print("Base");
    }
}
public class Alpha extends Base
{
    public static void main(String[] args)
    {
        new Alpha(); /* Line 12 */
        new Base(); /* Line 13 */
    }
}
```
- Select all queries that return number of students whose first name is John.
```
TABLE student
  id INTEGER NOT NULL PRIMARY KEY
  firstName VARCHAR(30) NOT NULL
  lastName VARCHAR(30) NOT NULL
```

Application:

- Fix the bug.
```
function average(a, b) {

        return a + b / 2;

}

console.log(average(2, 1));
```
- A palindrome is a word that reads the same backward or forward.

  Write a function that checks if a given word is a palindrome. Character case should be ignored.

For example, isPalindrome("Deleveled") should return true as character case should be ignored, resulting in "deleveled", which is a palindrome since it reads the same backward and forward.

- Binary search tree (BST) is a binary tree where the value of each node is larger or equal to the values in all the nodes in that node's left subtree and is smaller than the values in all the nodes in that node's right subtree.

  Write a function that checks if a given binary search tree contains a given value.

  For example, for the following tree:

  ```
  n1 (Value: 1, Left: null, Right: null)
  n2 (Value: 2, Left: n1, Right: n3)
  n3 (Value: 3, Left: null, Right: null)
  ```
  Call to contains(n2, 3) should return true since a tree with root at n2 contains number 3.
- Write a function that, given a list and a target sum, returns zero-based indices of any two distinct elements whose sum is equal to the target sum. If there are no such elements, the function should return null.

  For example, findTwoSum(new int[] { 1, 3, 5, 7, 9 }, 12) should return a single dimensional array with two elements and contain any of the following pairs of indices:

  - 1 and 4 (3 + 9 = 12)
  - 2 and 3 (5 + 7 = 12)
  - 3 and 2 (7 + 5 = 12)
  - 4 and 1 (9 + 3 = 12)
- Implement a function folderNames, which accepts a string containing an XML file that specifies folder structure and returns all folder names that start with startingLetter. The XML format is given in the example below.

  For example, for the letter 'u' and an XML file:

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <folder name="c">
      <folder name="program files">
          <folder name="uninstall information" />
      </folder>
      <folder name="users" />
  </folder>
  ```
  the function should return a collection with items "uninstall information" and "users" (in any order).
- Implement function countNumbers that accepts a sorted array of integers and counts the number of array elements that are less than the parameter lessThan.

  For example, SortedSearch.countNumbers(new int[] { 1, 3, 5, 7 }, 4) should return 2 because there are two array elements less than 4.
- Implement the removeProperty function which takes an object and property name, and does the following:

  If the object obj has a property prop, the function removes the property from the object and returns true; in all other cases it returns false.
- Write a function that provides change directory (cd) function for an abstract file system.

  Notes:

  Root path is '/'.
  Path separator is '/'.
  Parent directory is addressable as "..".
  Directory names consist only of English alphabet letters (A-Z and a-z).
  For example, new Path("/a/b/c/d").cd("../x").getPath() should return "/a/b/c/x".

  Note: Do not use built-in path-related functions.

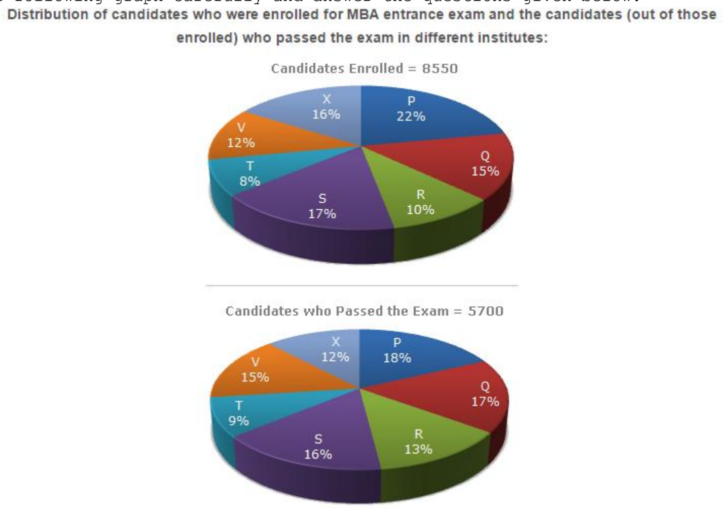
- A TrainComposition is built by attaching and detaching wagons from the left and the right sides.

  For example, if we start by attaching wagon 7 from the left followed by attaching wagon 13, again from the left, we get a composition of two wagons (13 and 7 from left to right). Now the first wagon that can be detached from the right is 7 and the first that can be detached from the left is 13.

  Implement a TrainComposition that models this problem.

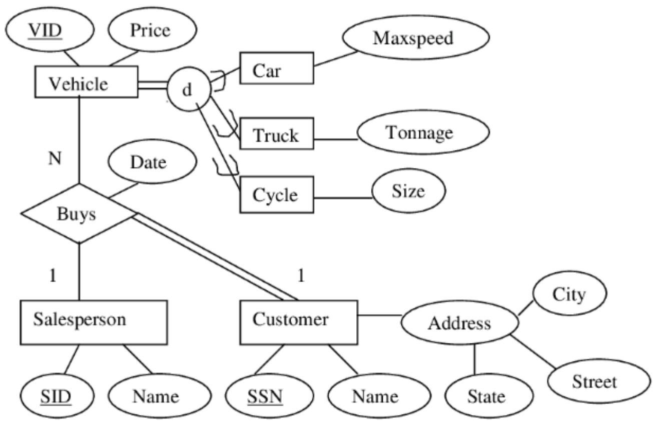
Analysis:

- Study the following graph carefully and answer the questions given below:



  1. What percentage of candidates passed the Exam from institute T out of the total number of candidates enrolled from the same institute?
  2. Which institute has the highest percentage of candidates passed to the candidates enrolled?
  3. The number of candidates passed from institutes S and P together exceeds the number of candidates enrolled from institutes T and R together by:
  4. What is the percentage of candidates passed to the candidates enrolled for institutes Q and R together?
  5. What is the ratio of candidates passed to the candidates enrolled from institute P?

- Consider the following ER diagram for the next two questions.



1. Which of the following statements is true?
    a. A Customer does not have to participate in the Buys relationship
    b. One Vehicle can be considered both a Car and a Truck
    c. A Customer buys a vehicle from one Salesperson
    d. A Customer is identified by the combination of Street, City and State
2. Which of the following statements is true?'
    a. A Customer can only buy vehicles from one salesperson
    b. A Salesperson can sell vehicles to only one Customer
    c. A given vehicle is sold by one Salesperson to one Customer
    d. All the above

Synthesis:

- Function appendChildren should add a new child div to each existing div. New divs should be decorated by calling decorateDiv.

  For example, after appendChildren is executed, the following divs:

```
  <div id="a">
    <div id="b">
    </div>
  </div>
```
  should take the following form (assuming decorateDiv does nothing):

```
  <div id="a">
    <div id="b">
      <div></div>
    </div>
    <div></div>
  </div>
```
  The code below should do the job, but for some reason it goes into an infinite loop. Fix the bugs.
- Fix the bugs in the registerHandlers function. An alert should display anchor's zero-based index within a document instead of following the link.

  For example, in the document below, the alert should display "2" when Google anchor is clicked since it is the third anchor element in the document and its zero-based index is 2.

```
<!DOCTYPE html>
<html>
  <head>
```

```
    <meta charset="utf-8">
    <title>Test page</title>
  </head>
  <body>
    In my life, I used the following web search engines:<br/>
    <a href="//www.yahoo.com">Yahoo!</a><br/>
    <a href="//www.altavista.com">AltaVista</a><br/>
    <a href="//www.google.com">Google</a><br/>
  </body>
</html>
```

- The createProductCodeForm function is used to create a new form that accepts a product code from a user.

  The current version of the form contains the hint: 'The product code can be found on the label'. This hint is currently always visible to the user.

  Improve the form so that the hint is only rendered when the input element is the focused element.

Evaluation:

-