Improved Programming Assignment Assessment with Grading Rubrics

CHRISTIAN LINDEBORG
Improved Programming Assignment Assessment with Grading Rubrics

CHRISTIAN LINDEBORG
Abstract

Oral assessment of lab assignments is a strong tradition in computer science courses at KTH and many other universities. The task of correctly assessing the students’ work tends to fall upon a teaching assistant (TA) who needs to be able to tell whether or not they actually understand the concepts covered by the lab. Previous research indicates that there is a large degree of uncertainty among TAs when performing this assessment, which can lead to unfair grades and uncomfortable situations for both TAs and students. This thesis presents an attempt to improve the situation through the creation of grading rubrics tailor made for three individual labs in computer science courses at KTH to be used during oral assessment. The grading rubrics created were preferred by the majority of TAs who partook in the study and seemed, as seen from their perspective, to improve fairness in assessment.
Sammanfattning

Muntliga redovisningar av laborationer i programmering och datalogi är en stark tradition på KTH och många andra universitet, där det faller på en labbassistent att bedöma studenternas arbete och avgöra om de har förstått de koncept som laborationen ämnar förmedla. Tidigare forskning har visat att assistenterna ofta är osäkra på hur de ska göra sina bedömnings vilket kan leda till orättvisa betyg och besvärliga situationer för både assistenter och studenter. Denna rapport förmedlar försöket att skapa bedömningsmallar skräddarsydda för tre laborationer i programmering och datalogi på KTH för att användas vid själva redovisningen. Bedömningsmallarna föredrogs av majoriteten av de tillfrågade labbassistenterna och ansågs av dem göra redovisningarna mer rättvisa.
Contents

1 Introduction 1
   1.1 Motivation, Sustainability and Public Welfare . . . . . . . 3

2 Background 4
   2.1 What is learning? . . . . . . . . . . . . . . . . . . . . . . . 4
      2.1.1 Constructive Alignment . . . . . . . . . . . . . . . . 6
   2.2 TAs in computer science . . . . . . . . . . . . . . . . . . 6
   2.3 Assessment . . . . . . . . . . . . . . . . . . . . . . . . . . 7
      2.3.1 Reliability and Validity . . . . . . . . . . . . . . . . 7
      2.3.2 Analytic and Holistic assessment . . . . . . . . . . 8
   2.4 Learning Taxonomies . . . . . . . . . . . . . . . . . . . . . 8
      2.4.1 Bloom’s Taxonomy . . . . . . . . . . . . . . . . . . 8
      2.4.2 2001 Revision of Bloom’s Taxonomy . . . . . . . . 10
      2.4.3 Taxonomy specific to Computer Science . . . . . . 12
   2.5 Grading Rubrics . . . . . . . . . . . . . . . . . . . . . . . . 12
   2.6 Descriptions of labs . . . . . . . . . . . . . . . . . . . . . 15

3 Methods 18
   3.1 Creation of Rubrics . . . . . . . . . . . . . . . . . . . . . . 19
      3.1.1 Preliminary interviews . . . . . . . . . . . . . . . . 19
      3.1.2 General Rubric Creation Methodology . . . . . . . . 20
   3.2 Usage of Rubrics . . . . . . . . . . . . . . . . . . . . . . . . 23
   3.3 Interviews with TAs . . . . . . . . . . . . . . . . . . . . . 23
   3.4 Questionnaire to students . . . . . . . . . . . . . . . . . . 25
   3.5 Ethical aspects . . . . . . . . . . . . . . . . . . . . . . . . 26

4 Results 27
   4.1 Iteration 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
      4.1.1 Finished rubrics . . . . . . . . . . . . . . . . . . . . 27
      4.1.2 Choice of criteria . . . . . . . . . . . . . . . . . . . 28
Chapter 1

Introduction

Computer science (CS) is a popular field of study. For example, in the United States the enrollment numbers for undergraduates within the field are at an all-time high [6] and in Sweden programming has recently been integrated into the math and technology curriculum for all elementary school students [15]. This popularity, in addition to the important role the field has in many companies and many parts of today’s society, means it is important to make sure the quality of CS education is at a high level.

Assessment of students’ work is an important part of that and also the topic of this thesis. How do we assess students in a way that is fair, precise and straightforward? Assessment of practical assignments (“labs”) is often done orally by teaching assistants of varying levels of experience. It is of course desirable for these assessments to not only correctly assess whether the students have understood the concepts in question but also that they are treated fairly, no matter which teaching assistant they interact with.

As a teaching assistant (TA) in a computer science course, it can sometimes be quite difficult to make assessments of a student’s understanding, when they present their work orally. This is especially true for beginner TAs but not limited to them [26]. There needs to be a solid basis on which the TAs can make their assessment. To fail a student, in the case of most TAs a fellow student, and someone they might have interacted with in other contexts, could be a sensitive thing. To choose, especially in a borderline case, to pass a student to avoid a tough situation could be tempting. It is often easier to pass than to fail [26].

Often the only thing a TA has access to in order to make the assess-
ment is a list of requirements that the students and their program are supposed to fulfill. These are often geared toward the practical side of things with the requirements on students’ understanding of the topic being more implicit. In the worst case there are no further instructions to the TA beyond the instructions of the lab itself. TAs may pick a somewhat holistic method of assessment, trying to get a bigger picture of the student’s understanding and coming up with questions on the spot that seem appropriate in the situation at hand, as opposed to going through a list of pre-defined questions. If happy and content TAs are desirable then this relative freedom to custom-tailor the assessment process can be a good thing, but can also lead to unfair assessments and uncertainty, especially for the more inexperienced TAs. However, if all students are asked the same list of questions, there might be an opportunity to abuse the system. Students may find out what these questions are, and prepare for them specifically, instead of aiming for a broader understanding of the topic.

In computer science courses, it is entirely possible for the student to have a perfectly functioning program that fulfills all the requirements of the assignment without actually understanding the concepts it is intended to convey, to a high enough degree [26]. Therefore the oral presentation of such assignments fills an important role, with the secondary benefit of making sure the students have the skills to explain their own work in a clear way, regardless of subject matter. It is of course desirable to make this process as clear and fair as possible. The feedback and personal interaction provided by the TA is also a good opportunity for the student to learn good programming style and learn to evaluate their own work as someone more experienced would. “Novice programmers are usually not very good at evaluating their own work, as even incorrect programs can seem to work as desired” [1].

The idea behind this thesis is to develop and evaluate a set of grading rubrics, a way of structuring the assessment and creating a common baseline for all TAs to follow. The questions I am posing are the following:

“Can teaching assistants in CS courses become more confident and fair in their assessment of programming assignments presented orally by students, by using grading rubrics designed specifically for the assignment in question? Can such rubrics
lead to a better assessment of students’ actual understanding of CS concepts?”

The work has taken place at KTH Royal Institute of Technology in Stockholm, Sweden. More specifically at the EECS school which hosts a variety of computer science courses. The courses in which the rubrics were used were both on a first cycle undergraduate level.

1.1 Motivation, Sustainability and Public Welfare

How can we judge whether this is a worthwhile goal? Without getting too philosophical we can reference the Global Goals for Sustainable Development[12] set out by the United Nations. Included among them is goal 4: Quality Education, where the aim is to “Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all”. Within the definition of that goal, we can find Target 4.7: Eliminate all discrimination in education, which is one of the things aimed at here. With clearer guidelines for how to assess students, there can be a higher degree of objective assessment. Raising the quality of CS and programming education in general, especially in these times, also applies to Target 4.4: Increase the Number of People with Relevant Skills for Financial Success.

The Standards and Guidelines for Quality Assurance in the European Higher Education Area (ESG), is also interesting, especially part 1.3: Student Centred Learning, Teaching and Assessment. Within it the following guideline stands out as especially applicable: Assessment is consistent, fairly applied to all students and carried out in accordance with the stated procedures [23].
Chapter 2

Background

A considerable amount of research has been conducted recently analyzing understanding among novice computer science students with some disappointing results [29] [19]. To quote Simon et al: “[…]students completing introductory programming courses can’t program at the level we expect they should”. Improved forms of assessment, especially of the formative kind that CS labs at KTH tend to lean towards, can be a way of alleviating this problem.

2.1 What is learning?

In order to have a solid foundation to stand on in answering the research question above, it is desirable to define what we actually mean by learning. When asking students what learning means, as many as six distinct views have been presented [18]:

- Increasing one’s knowledge
- Memorize and recall
- Take in knowledge and apply it
- Understand
- See something from a new point of view
- Change as a human being
There are a number of well defined theories with their own view on what learning actually means [22]. An early contender is that of behaviorism, introduced by John B Watson. The theory focuses on stimuli and reinforcement, illustrated most famously by the experiments done by Ivan Pavlov on dogs, which were conditioned to associate the sound of a bell with food being served, to the point were they began to salivate when hearing it. In behaviorism, learning is seen as a conditioned behaviour. Learning is adjusting behaviour according to signals from the environment. Whether this tells the whole story of how humans learn has of course been disputed by subsequent theories. If all we do is respond to stimuli, do we actually make any meaningful choices of our own? And can we generalize from experiments done with animals to learn about our own, human learning?

One such competing theory is the cognitive constructivism set forth by Jean Piaget [20]. It focuses on what goes on within the learner, and the model of reality that takes shape there. There is the concept of schemata, mental connections between concepts, on an appropriate level of detail. It’s a type of abstraction that potentially hides a lot of underlying complexity. When something new is learned it’s either assimilated into the existing network of schemata, fitting in alongside that which the individual already knows, or, if it is contradictory to that learned previously, it is accommodated, requiring a restructuring of said schemata, thus changing the individual’s mental model of the world, much like a revolutionary scientific discovery would invalidate some earlier theories.

Social constructivism is one further theory, of which one of the foremost thinkers was Lev Vygotsky [28]. Here the context of the learning is in focus, more so than the individual. Learning takes place not within the learner, and not in a vacuum, but in the interaction between individuals. An important concept here is the zone of proximal development (ZPD). It states that for all individuals, there are certain tasks within their ZPD, that they are able to perform on their own. But when they venture further, outside their ZPD, they don’t yet know enough to perform the task so on their own, but can do so with the help of someone more skilled at that particular task.

Cognitive constructivism makes for a good foundation in my own opinion when reasoning around learning and how to best go about it. And luckily enough, there is a well established framework for how to put it into practical use, described next.
2.1.1 Constructive Alignment

Constructive alignment is a relatively recent and popular principle used to construct teaching and learning activities, along with the associated assessment [17]. A key element is what is known as Intended Learning Outcomes (ILOs). The name comes from the combination of two concepts:

- A constructivist theory of learning. Which, like described above, means that learners use their own activity to construct their knowledge as interpreted through their own existing schemata.

- Alignment, which in this case is a principle in curriculum theory which means that teaching and assessment tasks should be aligned with the ILOs.

The ILOs are the starting point in constructive alignment. What is it the learner is supposed to learn? When those outcomes are established, teaching and assessment are aligned to them. The ILOs also follow a specific format, always containing a learning activity in the form of a verb. A verb that tells us something about what the learner needs to do to achieve the ILO, for example explain, apply or construct. Some possible examples from the domain of programming and CS:

- Explain collision-handling in a hash table
- Create a function that returns another function
- Analyze the complexity of a given function

Learning is in other words achieved through what the learner does, not what the teacher does. And when it comes to assessment from a constructive alignment point of view, it’s all about assessing how well students achieve those outcomes, not that they can repeat something they’ve been told in a lecture.

2.2 TAs in computer science

Within computer science at KTH, a TA is someone who assists the course responsible teacher during a course with potentially several things. Most TAs work during the lab sessions in the computer labs,
where they both help students with their assignments and assess their work when they’re done, deciding whether to pass them or not. Some TAs are also responsible for holding workshops and smaller lectures.

When assessment is made during a lab session the TA sits down with what is usually two students (lab assignments are most often done in pairs, or by a single student) and asks a number of questions to them in order to assess whether they’ve achieved understanding of the concepts that assignment was meant to convey. The code is usually run as well during this time, to ensure that the program fulfills all the requirements of the assignment. Most of the time both students naturally take turns answering questions, but if not the TA needs to direct questions to individual students, making sure they both show enough understanding to pass. There are usually several TAs working at once and the students have no control over which one they present their work to.

An early inspiration for my topic was the research done by Emma Riese at KTH, who interviewed a number of TAs working in entry-level programming courses about their experiences. An important conclusion of her work was the fact that many TAs, even the more experienced ones, felt a certain degree of uncertainty when assessing assignments presented by students [26]. These uncertainties were mainly concerning assessing the students’ actual understanding of the concepts, as opposed to the functionality of the program itself. The guidelines provided were often lacking when it came to the concepts the students were to learn, instead focusing mostly on the practical requirements of the assignment.

2.3 Assessment

Assessment in education can be described as “the systematic process of documenting and using empirical data on the knowledge, skill, attitudes, and beliefs to refine programs and improve student learning” [2]. Or alternatively as “getting to know our students and the quality of their learning” [24].

2.3.1 Reliability and Validity

How do we judge the quality of assessments? Two important terms that are of use to us are reliability and validity. An assessment with
high levels of both can in general be said to be of high quality.

Reliability describes how consistent an assessment is. If the results of an assessment vary wildly between different groups of students (who should reasonably be of similar skill level) or different times that an assessment was made, it shows signs of low reliability.

Validity on the other hand describes how well an assessment measures what it is actually supposed to measure. Judging someone’s ability to speak English through a written test for example, would be an assessment of quite low validity.

It should be noted that it is the combination of these two concepts that is the goal. An assessment can be quite reliable without being valid and vice versa. For example a quiz that always gives all students zero points is extraordinarily reliable, but probably not very valid, and certainly not that useful.

2.3.2 Analytic and Holistic assessment

The different methods of assessing someone’s work and deciding upon a grade can, according to Sadler [27], be done in two ways, analytic or holistic. Analytic assessment means we have a number of criteria that are individually judged and a final grade is calculated according to some sort of formula, while holistic assessment attempts to assess things as a whole.

2.4 Learning Taxonomies

A learning taxonomy (also known as an educational taxonomy) is a way to classify and define different kinds of learning and knowledge, and can be of help in the development of course curricula or, in this case, assessment guidelines.

2.4.1 Bloom’s Taxonomy

The first of its kind, Bloom’s Taxonomy, named after Benjamin Bloom who led the development of it, concerns itself with three different domains, of which the cognitive domain was the first described, and also the one that applies here [4].

In the cognitive domain it is divided into six different levels of objectives:
Knowledge
Comprehension
Application
Analysis
Synthesis
Evaluation

They are ordered in terms of complexity where knowledge is the most basic level consisting of memorizing terminology etc, while on the other end of the spectrum evaluation means the student has mastered the subject enough to really reason about different solutions to problems and evaluate how good they are.

The programming labs at KTH are in general pass or fail only, there is no graded assessment of them. The completion of the assignment itself, creating a functioning program that fulfills all the requirements, mostly demonstrates objectives at the levels of application and synthesis. One can see the process of presenting the work orally to a TA and explaining one’s thinking as confirming to the TA that the code was written by, and is understood by, the student, thus demonstrating learning on the level of knowledge, comprehension and perhaps also analysis and evaluation, depending on the nature of the questions asked.

Starr et al. recommend using Bloom’s taxonomy for exploring, specifying and refining assessable learning objectives in CS courses [30]. They emphasize the advantage of using the same process and terminology across the whole faculty to improve communication between educators and improving the consistency of students learning experiences. The process they used can be summarized simply as going through each topic that is to be covered in a course, and then, looking at the different levels in the taxonomy, in order from simple to complex, decide which level the students should be at after completion of the course. In using this process over a period of several years, a phenomenon they named concept shifting was observed. The meaning behind the term is that the concept being considered is inadvertently changed to a more simple one during the process of assessing its taxonomy level. The example used by the authors is that of when considering the concept of iteration, as opposed to the less abstract concept of
Figure 2.1: Bloom’s taxonomy with higher application added by Johnson and Fuller

a for-loop. On the synthesis level for example, if considering a for-loop the issue is simply to write a functioning for-loop, while if considering the more abstract concept of iteration, it would require the student to create their own version of iteration, a much more complex issue.

Johnson and Fuller however argued that the use of Bloom’s taxonomy in computer science was problematic because of the strong focus that’s often placed upon application, doing actual programming [13]. Educators being interviewed in the study felt that the two higher levels of Bloom’s taxonomy - synthesis and evaluation - were not really that applicable, especially during the first years of a CS program, and that application, a lower level in the taxonomy was the “core” of what computing is about. The authors suggested introducing a seventh level at the top called higher application (see figure 2.1), by which they mean “…application informed by a critical approach to the subject, but where the criticism is not, as such, the focus of the work”, contrasting it to the focus that is often seen in the social sciences.

Observing that the lower levels of Bloom’s taxonomy tended to be overlooked in modern CS education, and the fact that students’ skill levels varied greatly, Lister and Leaney successfully created a computer science grading scheme based on the taxonomy with the goal of effectively teaching and challenging students of all levels [16].

2.4.2 2001 Revision of Bloom’s Taxonomy

After almost 50 years of usage, a revision to Bloom’s taxonomy (in the cognitive domain) saw the light of day. Anderson et. al. restructured it
two-dimensionally along two dimensions: Knowledge (4 categories), and Cognitive Processes (6 categories), resulting in 24 possible slots in the table [14]. Less emphasis is placed on the hierarchical structure than in the original.

<table>
<thead>
<tr>
<th>Knowledge Dimension</th>
<th>Remember</th>
<th>Understand</th>
<th>Apply</th>
<th>Analyze</th>
<th>Evaluate</th>
<th>Create</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Factual</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Conceptual</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Procedural</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Metacognitive</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Knowledge Dimension

The knowledge dimension is divided into the following four categories:

A **Factual knowledge** - The basic elements that students must know to be acquainted with a discipline or solve problems in it. Terminology.

B **Conceptual Knowledge** - The interrelationships among the basic elements within the larger structure that enable them to function together.

C **Procedural Knowledge** - How to do something; methods of inquiry, and criteria for using skills, algorithms, techniques, and methods.

D **Metacognitive Knowledge** - Knowledge of cognition in general as well as awareness and knowledge of one’s own cognition.

The Cognitive Process Dimension

The cognitive dimension is divided into six categories. The number is the same as the original taxonomy but the names have changed from nouns to verbs and two categories have switched places:

1. **Remember** - Retrieving relevant knowledge from long-term memory.
2. **Understand** - Determining the meaning of instructional messages, including oral, written and graphic communication.

3. **Apply** - Carrying out or using a procedure in a given situation.

4. **Analyze** - Breaking material into its constituent parts and detecting how the parts relate to one another and to an overall structure or purpose.

5. **Evaluate** - Making judgments based on criteria and standards.

6. **Create** - Putting elements together to form a novel, coherent whole or make an original product.

Focusing on only the cognitive process dimension, the taxonomy is summarized in a simpler way in Table 2.1, in a similar fashion to one by Elmgren et al [10].

A list of examples of how to interpret the different levels in this revised edition of the taxonomy was compiled by Thompson et al [31].

### 2.4.3 Taxonomy specific to Computer Science

Despite the popularity of Bloom’s Revised taxonomy there are other potential alternatives. Could there be educational taxonomies that are a better fit with the field of computer science? Fuller et al. came to the conclusion that a major weakness of Bloom’s taxonomy is that “[its] levels do not appear to be well ordered when used to assess practical subjects such as programming” [11], and instead suggest a model where the six levels are separated into two dimensions, creating a matrix where one can identify more than one learning trajectory.

### 2.5 Grading Rubrics

**What is a grading rubric?**

A grading rubric is “a scoring guide used to evaluate the quality of students’ constructed responses” [21]. It is usually presented as a table, with every row pertaining to something that needs to be scored.
<table>
<thead>
<tr>
<th>Level</th>
<th>Step</th>
<th>Description</th>
<th>Verb</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Create</td>
<td>Put elements together to form a coherent or functional whole; reorganise elements into a new pattern or structure</td>
<td>Generate, list hypotheses. Plan, design. Produce, compose.</td>
</tr>
<tr>
<td></td>
<td>Evaluate</td>
<td>Make judgments based on criteria and standards.</td>
<td>Control, coordinate, demonstrate, map, try out. Criticise, make judgements.</td>
</tr>
<tr>
<td></td>
<td>Analyse</td>
<td>Break material into constituent parts and determine how parts relate to one another and to an overall structure or purpose.</td>
<td>Separate, distinguish, characterise, focus, choose. Organize, observe links, integrate, account for fundamental features, perform text analysis, structure. Attribute, deconstruct.</td>
</tr>
<tr>
<td>Low</td>
<td>Apply</td>
<td>Carry out or use a procedure in a given situation</td>
<td>Execute, perform. Realise, use.</td>
</tr>
<tr>
<td>Simple</td>
<td>Understand</td>
<td>Construct meaning from instructional messages, including oral, written and graphic communication.</td>
<td>Interpret, clarify, rephrase, translate, reproduce. Exemplify, illustrate, classify, categorise, arrange. Summarise, abstract, generalise. Deduce, conclude, extrapolate, interpose, foresee, compare, contrast, map out, match. Explain, construct models.</td>
</tr>
<tr>
<td></td>
<td>Remember</td>
<td>Retrieve relevant knowledge from long-term memory.</td>
<td>Recognise, identify. Recreate, recall.</td>
</tr>
</tbody>
</table>

Table 2.1: Bloom’s revised taxonomy
Grading rubrics actually come in many forms and different people have different ideas of what they are. A specific type of rubric might be prevalent in a certain academic institution for example, whereas the term has a different meaning elsewhere.

A much cited early article on rubrics by W. James Popham [21] lists three essential parts that make up a rubric: evaluative criteria, quality definitions and a scoring strategy.

Rubrics can vary in different ways and in an overview by Phillip Dawson, 14 different design elements/dimensions were identified (with three of them being the ones identified by Popham) in order to create a common language for the discussion of different types of rubrics [7]. The 14 dimensions are:

- **Specificity**: Is the rubric designed for a specific task or is it more general?
- **Secrecy**: Who gets access to the rubric? Everyone or just the person making the assessment?
- **Exemplars**: Work samples to illustrate quality.
- **Scoring strategy**: How is the final grade/score calculated? Analytic or holistic or something in-between?
- **Evaluative criteria**: Criteria that are used to distinguish acceptable responses from unacceptable responses [21].
- **Quality levels**: Different levels of quality achievable by the student. Normally in the form of column headers in the rubric.
- **Quality definitions**: Criteria which have to be fulfilled for an attribute at a given level.
- **Judgement complexity**: The level of expertise required by the users of the rubric.
- **Users and uses**: Who uses the rubric and how?
- **Creators**: The designer of the rubric. Could be a teacher, text book author, the students themselves or a combination.
- **Quality Processes**: Processes to validate rubrics or test their reliability.
• **Accompanying feedback information:** Feedback to the student included in the rubric itself.

• **Presentation:** The layout and format of the rubric. Is usually a grid with blocks of text but there are many possible variations.

• **Explanation:** Additional information presented alongside the rubric explaining how to make use of it.

There have been a number of studies indicating that rubric use can “lead to a relatively common interpretation of student performance” [25]. However, it is of utmost importance that the rubrics used are constructed with care, using appropriate and clear language [25].

### Grading Rubrics in Computer Science

Are rubrics a good fit for practical lab assignments in computer science? To judge if they are, a good sign would be if others have done it successfully, and there are examples of it.

Becker [3] for example, successfully used rubrics in programming courses, both for specific assignments and as a tool for assessing style and design more generally. For every lab there was one rubric specifically tailored to that assignment’s requirements, and one separate rubric for style, which was the same for every lab in the course, so students had a chance to get familiar with it. The rubrics were in a 2D grid, where each row described one part of the program, problem or solution and every column represented a level of achievement. The rubrics, complete with how many points were awarded for each part, were available for students to look at beforehand so they could judge how much effort they would need to put into the assignment.

### 2.6 Descriptions of labs

In order to evaluate the usage of the rubrics there needed to be some actual lab assignments to use. The ones chosen, and described below, are from the two courses *Applied Programming and Computer Science* (course code DD1321, course offering tilpro18), and *Applied Computer Science* (course code DD1320, course offering tildav19). The courses are both aimed at engineering undergraduate students at KTH who are not majoring in computer science, and have taken some other previous
course in programming. The intended learning outcomes (ILOs) of these two quite similar courses that are most relevant to the labs chosen are as follows:

- Use abstraction as a tool to simplify programming [9] (Lab P2 in course DD1321, in the rest of this report referred to as Lab 1).

- Compare algorithms with respect to time and memory usage [8](Lab 6 in course DD1320, in the rest of this report referred to as Lab 3).

- Describe different algorithms for searching and sorting and their properties [8](Lab 6 in course DD1320, in the rest of this report referred to as Lab 3).

- Implement and use hash tables and hash functions [8] (Lab D2 in course DD1321 and Lab 7 in course DD1320, collectively referred to as Lab 2 in the rest of this report).

The original instructions for the labs (in Swedish) can be found in the appendix. Here follows a brief summary of each one:

**Lab 1 - Functions**

Lab 1 centers around creation of functions and using them for abstraction. Among other things a function that performs multiplication by internally just performing a number of additions. The algorithms created need to be explained with flow charts and analyzed in terms of time complexity.

**Lab 2 - Hash Tables**

Lab 2 is about hash tables and their implementation. Two different versions have to be created, both implementing the same abstract data type. Collision handling, optimal table size and what to consider when creating a hash function are among the details covered.

**Lab 3 - Searching and Sorting**

Lab 3 requires students to perform time complexity analysis and measure the actual time taken to perform linear search, sorting (Quicksort or Mergesort), binary search and a hash table lookup. The calculated
time complexities and actual times are then to be compared and discussed.
Chapter 3

Methods

In order to be able to confidently answer the research questions at hand, let us first remind ourselves of what those questions were:

“Can Teaching Assistants in CS courses become more confident and fair in their assessment of programming assignments presented orally by students, by using grading rubrics designed specifically for the assignment in question? Can such rubrics lead to a better assessment of students’ actual understanding of CS concepts?”

To answer these questions, sets of grading rubrics were created for three different lab assignments, spread out over two courses, Applied Programming and Computer Science (tilpro18) and Applied Computer Science (tildav19). As the names imply, these courses have some similarities and one of the labs chosen appears in both courses, which means during each course two rubrics can be used. Having both of these courses available to me meant that there was an opportunity to revise the rubrics in between if needed, creating two different iterations of the experiment. The details of the labs were described in section 2.4.

The following is a simple timeline of events:

<table>
<thead>
<tr>
<th>Table 3.1: Timeline of Iterations and labs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Iteration 1</strong></td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>tilpro18</td>
</tr>
<tr>
<td>Lab 2</td>
</tr>
</tbody>
</table>

18
• Lab 1: Functions (fall only)
• Lab 2: Hash Tables (fall and spring)
• Lab 3: Searching and Sorting (spring only)

It should be noted that I myself worked as a TA in both these courses during the experiment, and had experience of doing so previously.

3.1 Creation of Rubrics

3.1.1 Preliminary interviews

To get a clearer picture of how TAs tend to experience the job, and to get some inspiration for what to focus on, as a first step, three informal interviews were conducted with TAs about their experiences working at KTH, focusing on the assessment of lab assignments. To get a wide array of viewpoints the degree of experience among them varied. One was quite new on the job, while the other two were very experienced. Some things that stood out in these interviews were:

• TAs are sometimes unsure of what answer is required of the students. It may be embarrassing to ask someone, and assessments can vary from TA to TA. Clearly having the correct answer available would increase the TAs confidence in making a judgment.

• Unless given clear instructions from the course responsible teacher, TAs themselves tend to try to identify which concepts are the important ones. They often pass students who have failed to understand something the TA sees as less important, but also make sure to tell them about it so that they hopefully learn about it then and there.

• Failing a student can be an awkward experience for both the student and the TA. Examples on when to fail a student should be as clear as possible to make the distinction easier. One TA who was new on the job had never failed anyone and had never seen another TA fail someone, so in his mind this seemed like a vary rare event.
3.1.2 General Rubric Creation Methodology

A important goal for rubrics, according to Popham, is for them to not be too lengthy, and efficient to use [21]. Reliability and transparency are also important aspects [25]. The following are ways to achieve this:

- Accurately assess students’ knowledge of the concepts in question, thus improving reliability.
- As clearly as possible define how to assess the work of students, thus contributing to both transparency and improved reliability.
- They should be relatively simple and fast to use, in other words, having a good efficiency.

My starting point was to go through the different dimensions defined by Dawson (see the previous chapter) to start shaping the rubric.

- **Specificity**: The rubrics need to be tailor-made in accordance with a specific lab-assignment. However the general design should be applicable to a variety of such assignments.

- **Secrecy**: The person making the assessment, the TA, is the intended reader. Why not the students? In many usages of rubrics the intended readership includes the students themselves, so that they can get a clear picture of what they need to achieve for a certain grade. In this case, the lab instructions should be quite enough for the students to know what is required of them, and the labs are all pass or fail. The individual criteria of the rubrics are expressed as questions. In order to avoid that students study only these specific questions instead of trying to achieve a wider understanding of the topic, the rubric is not intended to be read by students.

- **Exemplars**: Providing samples of work that fulfills the requirements could make it clearer how to make a correct assessment, but it would take up a lot of space, especially perhaps if some of the exemplars should be in the form of code. Since an important goal here is to have the rubric be simple and fast to use I have chosen to not use them.
- **Scoring strategy:** The final score for a CS lab assignment at KTH is in the vast majority of cases a simple pass or fail. Certain requirements of the program need to be fulfilled and certain theoretical concepts need to be understood in order to pass. The rule for calculating the final score simply becomes: If the students have produced a program that fulfills all the requirements specified in the assignment, and they show in a satisfactory way that they’ve understood the theoretical concepts at hand, they pass. Otherwise they do not.

- **Evaluative criteria:** One of the main problems that are hopefully solved by these rubrics is specifically this. How do the TAs know which answers are acceptable and which are not? Despite being able to solve problems in many different ways, solutions tend to end up in a limited number of categories and these can be represented using a column with the most typical solution and an additional column with comments on other possible solutions. This could also contain information on solutions that are not acceptable of course.

- **Quality levels:** Since the labs are assessed on a pass or fail basis it’s tempting to say that we simply don’t need any differentiation of quality levels. However, some concepts are more important than others. Some are essential, and the students need to be able to show that they truly have an understanding of it to pass. In other situations it can be entirely alright to treat it as a more formative criteria, we’re not done until we’ve covered it and discussed it. In this case it is acceptable for the student to have made some mistakes, and for the TA to fill in the knowledge gaps for them. The role of the rubric in this case is to clearly show which is which. A way of representing this in a rubric is to have four columns, “OK”, “Slightly wrong”, “Wrong” and “Not applicable”, which for every question/criteria is filled in with a color showing how harshly it should be judged with the colors being:
  - **Green:** OK
  - **Yellow:** Small error. Ask the students to fix it and come back later during the same lab session.
  - **Red:** NOT OK
– Black: Not applicable

• **Quality definitions:** Assessing a criteria as “OK” or “Wrong” can be covered by the column containing the expected reply and the one with comments (see Evaluative Criteria). The question is then how do we distinguish something as “Slightly wrong”? This could also be covered by the previously mentioned columns, but there’s also room in the “Slightly wrong”-column itself to insert examples of these errors.

• **Judgement complexity:** The users of the rubric are expected to be familiar with the CS concepts being assessed.

• **Users and uses:** The intended users are TAs, who might have taken the course in question themselves or are at least familiar with the concepts from similar courses. It’s to be used when students orally present their work on a lab assignment. As mentioned previously, the intention is not that students should be able to take part of the rubric in its entirety beforehand, but for the sake of transparency and clarity the TA might need to show the rubric to the student during assessment.

• **Creators:** In this case the rubrics are obviously created by me, as part of this thesis, and as a form of research, but if the format is found to be a useful tool, the intended creator of each specific rubric is the teacher, perhaps along with the TAs.

• **Quality Processes:** This has two different interpretations, one being how I, during the course of my research will validate the rubrics and test their reliability, and the other being how a potential user of the format in the future would do so. In the first case, I’ve described later in this chapter how interviews were conducted in order to evaluate the rubrics and the general format after being used. When it comes to future users, creating their own specific rubrics using the format, I have no further suggestions than for the creator to discuss them with the TAs using them before, during and after usage.

• **Accompanying feedback information:** The columns for “Wrong” and “Slightly wrong” can contain feedback information.
• **Presentation:** The rubric needs to be simple and fast to use. A standard grid format with text and with some fields colored in for clarity when assessing criteria seems appropriate. The goal will also be to fit the entire rubric for a given lab onto one single sheet of paper, that can be handed out to TAs at the lab session.

• **Explanation:** To keep the rubric as uncluttered and simple as possible, any additional information considering how to make use of it will be presented separately, as written instructions available to the TAs, and / or as workshop or small seminar with the TAs at the beginning of the course.

### 3.2 Usage of Rubrics

Three sets of rubrics, one for each lab above, were created and made available to the TAs working on the course in question. For the first iteration, they were made available to the TAs through a university website accessible by the course staff, as well as handed out in printed form at the lab sessions where the focus was on the labs in question. The idea was that the TAs would be able to look at the rubric beforehand but that they should always have access to it in paper-form at the labs. I was present myself to hand them out on most occasions. When I was not available, I delegated this to another TA who was present.

### 3.3 Interviews with TAs

In order to evaluate the rubrics’ usefulness, several interviews were carried out with TAs that had used them. This was done in two separate iterations, once for each course. In both cases, it was made sure that the interviewees had time to experience both using the rubrics, as well as working during lab sessions where labs *without* rubrics developed for them were being presented, so that even the newest TAs had experience of both.

The interviews were in a semi-structured format [5], with prepared questions about the following topics:

• Level of experience working as a TA, including whether they had been a TA for the course in question previously.
• Comparison between using the rubric, and using the instructions they otherwise receive as a TA, in the following aspects:
  – Confidence in your assessment
  – The students understanding of the topic
  – Not missing important parts
  – Assessment matching that of other TAs
  – Reactions from students to questions asked
  – Time taken to do the assessment

• If they had any suggestions on how the rubric could be improved.

• If they had received sufficient information about the rubrics and how they should be used.

• Finally, would they choose to use a rubric such as the one they used or the type of instructions they otherwise would get, if they had the choice?

In iteration 1, 5 different TAs of varying levels of experience were interviewed and some of the feedback from said TAs was used to refine the rubrics for the second iteration.

In the second round of interviews (iteration 2), a question was added asking whether the interviewee had been present at the meeting with the TAs where the intended usage of the rubrics was discussed.

The results were then analyzed through the method of thematic analysis, a popular form of qualitative analysis. It can be summarized a method for identifying, analysing and reporting patterns (themes) within data [5]. I was not familiar with analyzing interviews before this and chose to use thematic analysis because it is a flexible, popular method intended for qualitative analysis that seemed simple enough to use.

A theme in this case captures something important about the data, in relation to the research question. The process for doing this type of analysis can be described by the following steps:

1. **Familiarize yourself with your data** - Immersing yourself into the data, and familiarizing yourself with it. This should be an active task, searching for patterns and meaning as you’re doing it. This step should also include any needed transcription of audio data.
2. **Generating initial codes** - With some familiarity of the material established and some ideas of themes and meanings, you can generate codes, simple labels that identify a feature of the data that seems interesting. Could be a single word of a short phrase that connects different fragment of the data that have a common meaning or relationship.

3. **Searching for themes** - A broader level of analysis than the codes, sorting them into themes. Go from a long list of codes to a more structured format. How can different codes combine to form overarching themes? This can be done with a mind map or other similar tool for categorizing into major themes and sub-themes.

4. **Reviewing themes** - Go over the candidate themes from the previous step and refine them. Perhaps some aren’t really cohesive themes? Perhaps two of the themes are essentially describing the same thing and could be combined into one? The opposite could also be true, that a theme is too wide and is better described as two separate themes.

5. **Defining and naming themes** - Identify the essence of each theme, what is interesting about them and why? What ‘story’ does the theme tell and how does it fit into the broader ‘story’ that the data tells us? Try to describe the scope and content of each theme in a couple of sentences. It’s also time to decide upon the names for the themes that will be used in the final analysis.

6. **Produce the report** - Tell the story of your analysis, extracts showing concrete examples illustrating the themes you describe. Make some sort of argument in relation to your research question.

### 3.4 Questionnaire to students

To get some data from the students’ point of view a simple open question was included in the course evaluation of the first course (tilpro18):

“During some of the labs (P2 - mult/div and D2 - hash-table) the TAs have been using special grading rubrics when assessing your work. Did the usage of these rubrics affect the experience of presenting your work? If so how?”
3.5 Ethical aspects

There are two main ethical aspects that I’ve taken into account during my work, how my experiments affect the students taking the course and how to treat the data gathered through interviews.

The students are simply taking a course like any other and expect to be fairly treated and to not be subjected to any unnecessary experimentation on risking the quality of their own learning. Of course the objective of my research is to have a positive effect on the way the course is taught but there’s no guarantee that will be the case. In order to as well as possible ensure that the students are not unfavourably affected by my work all rubrics used and the methods for using them were discussed thoroughly with the course responsible teacher.

When it comes to the interview data, every interviewee was informed that the data and any eventual quotes would be presented in an anonymous fashion, so that they could not be individually identified, and that they only need to answer the questions they were comfortable answering.
Chapter 4

Results

4.1 Iteration 1

In iteration 1 the initial rubrics (Lab 1 and Lab 2) were designed as well as the general format. The changes in iteration 2, that are described later, were relatively small so most of the work was concentrated in this first iteration.

4.1.1 Finished rubrics

The rubrics used in practice were all in Swedish and one of them (Lab 2 - Hash tables) has been translated here to explain in more detail how they were created. All original rubrics in Swedish can be found in the appendix.

The layout and general appearance of the tables is based on the considerations made in the previous chapter around the 14 different dimensions described by Dawson, as well as an ambition to make the levels of reliability and validity of the assessment high.

Every individual row/rubric contains a question, that the TA should ask the student, which in a way summarizes what that specific rubric is about. The column with learning outcomes is meant to supplement this. The choice of which questions to actually include and how to assess them was based primarily on the learning outcomes and instructions provided in the lab instructions themselves. I go into more detail on the design of individual criteria in the next section.

There was also the consideration of different levels of learning according to the revised version of Bloom’s taxonomy. I chose to use
this specific taxonomy as a reference point instead of other alternatives mostly because of its wide acceptance in the field. As can be seen, there is a column describing what level a question assesses. This did not actually need to be present on the rubric used in practice, an oversight on my part, but in retrospect it illustrates the process of choosing the questions. I wanted to make sure both student A and B were asked questions on varying levels. This is to ensure a high reliability. So the ordering of the questions, along with the first column that shows which student should be asked the question, was set in a way that gave a good spread over the different levels and over the different subjects covered in the lab.

There is also some sort of priority in the ordering, in that the uppermost rows are just to see that the program written actually functions as it is meant to, while the lowermost row is a discussion question, the students don’t actually need to be able to answer it, but it can be an interesting and formative topic to discuss if there is time.

In some questions where there is perhaps more than one simple answer additional info is available in the comment column.

If students question the rubrics, the last column is a sort of summary of the info that the students actually received in the lab instructions.

4.1.2 Choice of criteria

We will keep looking at Lab 2 (hash tables) as an example on how the individual criteria and questions were chosen and formulated.

“What do we get back when we put an element with key x in the hash table and then search for the key x?”

In order to check that the program created actually does what it’s supposed to, and to not waste time on an assessment that won’t pass anyway, the first two questions are there to check if the program behaves correctly. As can be seen in the student column, they are not directed at any particular student, they are just there to confirm that the assignment seems to have been completed. One fundamental property of a hash table, or at least the interface of one, is that if we put a value into it, with a certain key, then getting the value of that key should produce the same value. And that is what these first two questions are
about. To be clear, these criteria only really check if what has been created has the correct interface, that it is an associative array. We need further questions to assess if it is actually a hash table under the hood. These questions, along with the third one, are also the only ones where even a slightly wrong answer puts us in the red, and the assignment is failed. This is to emphasize their importance, and so that the assistant can quickly move on to other assessments while the students get on with fixing the flaws in their program.

“Why is it faster to find an element in a hash table than a regular list?” / “What is the time complexity of finding an element in a hash table with n elements if we disregard collision handling?”

Moving more into the theory of hash tables, we need to know if the students understand the inner workings of them. A fundamental property of any hash table, and the main reason they are such an important data structure, is their speed, and a student who explains why a hash table is as fast as it is, naturally has to explain how it works. That there is a hash function for example, and that it is only runs once regardless of the table’s size. In this question we ask them to compare this speed with that of lists, a data structure they are already familiar with.

The fourth question concerns itself with pretty much the same thing, but focuses on the time complexity. They need to show that they can relate the practical behaviour they just described to the more theoretical concept of time complexity.

“What properties should a good hash function have and why?” / “How does your hash table work?”

Without a good hash function the advantages of a hash table are not as great anymore, and as mentioned previously, we need to look closer to see if what the students have actually created is a proper hash table. The fundamental properties, necessary for it to even work at all are included in the expected answer, but the focus here is instead on the spread, what can make one hash function better than another. For example, if it isn’t obvious to the students that a hash function that always returns the same value is a very bad one, they have not understood the role of the hash function.
“What do we mean by a collision in a hash table?” / “What method for handling collisions are you using and how does it work?”

Another double check to see that the concept of the hash table is clear to the students. If they don’t understand that collisions have a chance of happening then they have probably not understood the concept fully, and might have constructed something that isn’t a proper hash table. It is unlikely but possible that they have created a perfect hash function that avoids collisions entirely. That would however be quite inefficient when it comes to the size of the table, and handling collisions are a clear part of the lab assignment. In the instructions for the lab the students are told to pick between using chaining or some kind of probing to handle the collisions, and that is mentioned in the expected answer column.

“Which size did you choose for your hash table and why?”

The important thing here is that they understand the need for room in the table, which is stated quite clearly in the expected answer and comment columns.

“Sketch a picture of your table and what happens as an element gets added. What does it look like when a collision occurs?”

As a sort of final check that the students have grasped the subject they need to draw the table. This is to ensure that they are not just repeating a practiced speech on how a hash table works. It is quite likely that they have spontaneously already done this as part of the previous questions, and then the assistant can skip this question (something I neglected to include in the rubric though).

“Considering the fact that the same artist sometimes appears several times in the list of songs, how does it affect the hash table if we choose to use the artist names as keys as opposed to using the song-IDs?”

To make the assessment a valuable experience for all students, even those who had an easy time doing the assignment, this more open question was included. There is no possibility to fail this criteria, instead it is a basis for further discussion.
A+B
Create hash tables

"What do we get back when we put an element with key x in the hash table and then search for the key x?"

"We should get the same value we originally put in back!"

Create their own implementation of a hash table

Test case: Add the string "Hi!" to the table with the key "message". Then search for the element with the key "message"

"We should get the element containing "Hi!" back."

They need to have a working hash table!

A
Explain data structure

"Why is it faster to find an element in a hash table than in a regular list?"

"The hash function is only run once, regardless of the size of the table."

Evaluate

Explain why a hash table enables fast lookup

B
Calculate time complexity

"What is the time complexity of finding an element in a hash table with n elements if we disregard collision handling?"

"O(1)"

Discuss how collision handling can affect this. What happens if everything is a collision and everything ends up in the same chain for example?

Analyse

See above

A
Create hash functions

"What properties should a good hash function have and why?"

"Return an index in the correct range. The same key should always give the same hash value. A good spread of values. Relatively fast."

Understand

Create their own hash function

B
Explain algorithm

"How does your hash function work?"

"Should in principle have the properties described above."

- They must have created their own hash function

A
Explain collision handling

"What do we mean by a collision in a hash table?"

"When two different keys result in the same hash value. The same key twice is not a collision. Should be handled differently."

Understand

Implement and explain collision handling with chaining or probing

B
Implement collision handling /

Explain algorithm

"What method for handling collisions are you using and how does it work?"

"Chaining or some kind of probing - They must have included collision handling"

Create

Implement and explain collision handling with chaining or probing

A
Create hash tables

"Which size did you choose for your hash table and why?"

"Double the size is a good rule of thumb, they should at least have understood that it needs to be a bit spacious. Having a prime or uneven size is not a priority here."

Evaluate

The hash table needs to be appropriately sized

B
Sketch data structure

"Sketch a picture of your table and what happens as an element gets added. What does it look like when a collision occurs?"

- -

Understand

Be able to sketch the hash table

A+B
Explain data structure

"Considering the fact that the same artist sometimes appears several times in the list of songs, how does it affect the hash table if we choose to use the artist names as keys as opposed to using the song-IDs?"

"We will get fewer actual entries in the table in the first case, when the same artist appears again the first appearance of that artist will be overwritten. Although their implementation might behave a bit different. If they don't overwrite the the older entry, but instead store the new one alongside the old ones, is there then any way of actually accessing more than one of those entries?"

Discussion-question

- Analyse

- OK

Needs to be completed (give them a while to work on it and come back later)

Fail

Not applicable
4.1.3 Iteration 1 interviews

A number of important and clear themes were identified after conducting the first round of interviews:

- *All* interviewees claimed they would choose the rubric format instead of the traditional instructions if they had the choice.

- *All* interviewees thought the rubric made assessments more fair by providing a clear baseline for what was required of the students.

- The information provided during the course about the rubrics, and the fact that they should even be used, was not sufficient, and many TAs were surprised to be handed a paper at the lab session with any instructions on how to use it being quite brief. Several interviewees suggested having a meeting with all TAs on the course to discuss the rubrics beforehand.

- The time taken to do the assessment, compared to when using traditional instructions, was largely unaffected.

Besides these, there were a number of suggestions and minor themes:

- The difference between the columns *wrong* and *slightly wrong* (and the yellow and red fields), was in practice not very useful. If there are things the students have missed, or not understood in a satisfactory manner, the TAs pretty much always give them the opportunity to correct this during the lab session, and if they manage to do so within that time the TA checks their work again. The important distinction is rather between something being good enough to pass, or not yet reaching that level.

- The column with Bloom-levels was still present on some of the printed out versions given to TAs. They had no idea what this was and it was slightly confusing.

- The rubrics made the assessment feel a bit more stale than usual.

- The rubrics made a direct difference for the TAs and their work. What impact it had on the students, directly or indirectly, is more unclear.
• Colors could be used more in the design of the rubric. For example, a difference in hue on the rows indicating which is student A and which is student B.

• Some questions were essentially about the same thing and could be combined into one.

• The rubrics had no clear title, meaning you had to look through the questions to see which lab it was about.

• The A/B-column was a good reminder to switch between students, even if it’s usually very hard to stick to the given order.

4.2 Iteration 2

For iteration 2, the TAs of the course were invited to a meeting by the course responsible teachers, where I presented the rubrics and how they were to be used. Any questions they had were answered so that things would be as clear as possible. Of the 5 TAs interviewed, 3 were present at the meeting.

4.2.1 Revised rubrics

For iteration 2 the rubric for the second lab was revised after feedback from the interviews in iteration 1, and a new rubric for lab 3 was created, using the same revised format. The changes made can be summarized as follows:

• Clear titles were added to each rubric, simply indicating which lab it described.

• The “slightly wrong” column was removed.
<table>
<thead>
<tr>
<th>Question</th>
<th>Expected answer</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expected answer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Comment</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Correct</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Incorrect</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Material missing</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Bloom-level</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Info to students</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Exercise</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Info to students</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Expected answer**

| **Expected answer**                                                     |                 |                                                                         |
| **Comment**                                                             |                 |                                                                         |
| **Correct**                                                             |                 |                                                                         |
| **Incorrect**                                                           |                 |                                                                         |
| **Material missing**                                                     |                 |                                                                         |
| **Bloom-level**                                                         |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |
| **Exercise**                                                            |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |

**Correct**

| **Expected answer**                                                     |                 |                                                                         |
| **Comment**                                                             |                 |                                                                         |
| **Correct**                                                             |                 |                                                                         |
| **Incorrect**                                                           |                 |                                                                         |
| **Material missing**                                                     |                 |                                                                         |
| **Bloom-level**                                                         |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |
| **Exercise**                                                            |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |

**Incorrect**

| **Expected answer**                                                     |                 |                                                                         |
| **Comment**                                                             |                 |                                                                         |
| **Correct**                                                             |                 |                                                                         |
| **Incorrect**                                                           |                 |                                                                         |
| **Material missing**                                                     |                 |                                                                         |
| **Bloom-level**                                                         |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |
| **Exercise**                                                            |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |

**Material missing**

| **Expected answer**                                                     |                 |                                                                         |
| **Comment**                                                             |                 |                                                                         |
| **Correct**                                                             |                 |                                                                         |
| **Incorrect**                                                           |                 |                                                                         |
| **Material missing**                                                     |                 |                                                                         |
| **Bloom-level**                                                         |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |
| **Exercise**                                                            |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |

**Bloom-level**

| **Expected answer**                                                     |                 |                                                                         |
| **Comment**                                                             |                 |                                                                         |
| **Correct**                                                             |                 |                                                                         |
| **Incorrect**                                                           |                 |                                                                         |
| **Material missing**                                                     |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |
| **Exercise**                                                            |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |

**Info to students**

| **Expected answer**                                                     |                 |                                                                         |
| **Comment**                                                             |                 |                                                                         |
| **Correct**                                                             |                 |                                                                         |
| **Incorrect**                                                           |                 |                                                                         |
| **Material missing**                                                     |                 |                                                                         |
| **Bloom-level**                                                         |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |
| **Exercise**                                                            |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |

**Exercise**

| **Expected answer**                                                     |                 |                                                                         |
| **Comment**                                                             |                 |                                                                         |
| **Correct**                                                             |                 |                                                                         |
| **Incorrect**                                                           |                 |                                                                         |
| **Material missing**                                                     |                 |                                                                         |
| **Bloom-level**                                                         |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |
| **Exercise**                                                            |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |

**Info to students**

| **Expected answer**                                                     |                 |                                                                         |
| **Comment**                                                             |                 |                                                                         |
| **Correct**                                                             |                 |                                                                         |
| **Incorrect**                                                           |                 |                                                                         |
| **Material missing**                                                     |                 |                                                                         |
| **Bloom-level**                                                         |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |
| **Exercise**                                                            |                 |                                                                         |
| **Info to students**                                                    |                 |                                                                         |
4.2.2 Iteration 2 interviews

Many of the same major themes made an appearance in this iteration:

- All interviewees thought the rubric made assessments more fair by providing a clear baseline for what was required of the students.

- All interviewees agreed that the time taken to do the assessment, compared to when using traditional instructions, was largely unaffected.

- All interviewees except one claimed they would choose the rubric format instead of the traditional instructions if they had the choice.

In contrast to the previous iteration, a majority of the interviewees thought that enough information about the rubrics had been provided. All who did had been to the information meeting while those who did not had not been at the meeting. The rubrics had not been made available online for the TAs to review beforehand, except for at the meeting, which those who were not at the meeting would have appreciated.

There were several minor themes and suggestions new to this iteration:

- The "Expected answer" column is a very good thing to have. It saves a lot of time especially for beginners.

- The rubrics make it a lot easier for a TA to make the jump between different courses and to work as a substitute in a course they’re not that familiar with.

- The colors red and green are not optimal from a color blindness perspective.

- It’s important to be able to clearly tell how important the different questions are, if some of them have higher or lower priority.

- The rubrics contain a lot of info and can seem cluttered. They could probably be streamlined even more.

- One interviewee claimed to prefer a simple list of requirements over the rubrics.
• Two interviewees complained that some details were missing in the rubric for the hash-table lab (Lab 2) concerning collision handling, that led to some differing assessments.

4.2.3 Questionnaire to students

For the course used in iteration 1 a question was asked to the students as part of the broader course evaluation:

“During some of the labs (P2, mult/div and D2 - hash-table) the TAs have been using special grading rubrics when assessing your work. Did the usage of these rubrics affect the experience of presenting your work? If so how?”

Only a minority of students (27 out of 193) provided answers to the course evaluation so it’s not possible to draw any certain conclusions. But the answers provided can however be summarized as follows (with some example quotes included):

• Several students claimed to not have noticed that the rubrics were being used. “No, I didn’t even notice.”

• Multiple students claimed it made the assessment seem more fair. “Without it it becomes unfair because some TAs ask other things than others depending on how knowledgeable they are and then someone less confident needs to just hope to get the right questions. That’s not how it’s supposed to be, same requirements for everyone!”

• Multiple students claimed it made the assessment process seem more structured. “It felt like a more cohesive assessment.”

• Multiple students claimed it might have made it easier to pass. “It made the assessment seem more fair, but at the same time I don’t think it required the same broad understanding in order to pass.”

• None of the students thought it had a negative impact.
Chapter 5

Discussion and Conclusions

The most obvious conclusions that can be drawn from the experiments and interviews are:

- The usage of the rubrics improved the fairness of assessments made by TAs, at least seen from the perspective of the TAs themselves.

- The majority of TAs would prefer to use rubrics such as the ones designed here instead of traditional instructions.

- The time required to perform the assessment is largely unaffected.

So can we answer the original research questions? Here they are again:

“Can Teaching Assistants in CS courses become more confident and fair in their assessment of programming assignments presented orally by students, by using grading rubrics designed specifically for the assignment in question? Can such rubrics lead to a better assessment of students' actual understanding of CS concepts?”

Concerning the first question, regarding fairness and clarity, I would say that the answer is yes. It can confidently be said that the rubrics are of benefit to the TAs. The second question is somewhat unclear. Confidence in the students’ understanding seems to not have been obviously affected by the use of rubrics, although some TAs stated that the rubric gave them a sense of confirmation, that they were asking the correct questions.
Based on these positive results, rubrics following the format described in this report could be beneficial to CS courses at KTH and elsewhere where work on practical lab assignments is assessed orally by TAs.

5.1 Possible improvements

Based on the responses from TAs there are a couple of improvements that could be made to the rubrics:

- Streamline further. For example the column describing the Bloom-levels still remained in most printed out rubrics. This was a mistake that should have been rectified for iteration 2 but wasn’t. The Bloom levels were a useful tool in the construction of the rubrics but serve no clear purpose for the TAs at the time of the assessment.

- Better usage of colors and patterns in the design of the rubric. Tone or color differences to make the Student A/B rows even more obvious for example. And perhaps other colors than red and green to help those with color blindness.

5.2 Future work

The focus of this thesis was more on the perspective of the TAs than the students. As mentioned previously an attempt was made to get some student input through a course evaluation but there were too few answers to draw any clear conclusions, although the answers that were provided were quite positive. Future work could focus more on the students and how they are affected.
Bibliography


Appendix A

Original Rubrics in Swedish

A.1 Iteration 1
<table>
<thead>
<tr>
<th>Student</th>
<th>Lärandemål</th>
<th>Moment / Fråga</th>
<th>Förväntat svar</th>
<th>Kommentar</th>
<th>Rätt</th>
<th>Lite fel</th>
<th>Fel</th>
<th>Underlag saknas</th>
<th>Bloom-nivåer</th>
<th>Info till Studenterna</th>
</tr>
</thead>
<tbody>
<tr>
<td>A+B</td>
<td>Skapa funktioner</td>
<td>&quot;Vad returnerar mult(3,4), mult(0,0) samt mult(3,-3)?&quot;</td>
<td>12</td>
<td></td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Skapa implementera multiplikation m.h.a plus och minus</td>
<td>A+B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A+B</td>
<td>Skapa funktioner</td>
<td>&quot;Vad returnerar div(12,3), div(13,3) samt div(3,0)?&quot;</td>
<td>4</td>
<td></td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Skapa implementera division m.h.a multiplikation</td>
<td>A+B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A+B</td>
<td>Förklara begreppen parameter och returvärde</td>
<td>&quot;Förklara begreppen parameter och returvärde&quot;</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Avser både anropsparametrar och de i definitionen. Diskutera båda.</td>
<td>A+B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A+B</td>
<td>Funktionsdelar</td>
<td>&quot;Hur skulle ni kunna skriva en funktion för att beräkna exponenter med hjälp av en funktion för multiplikation?&quot;</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Funktionsdelar</td>
<td>Analysera</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A+B</td>
<td>Kommentarer</td>
<td>&quot;Har ni kommenterat era funktioner? Varför är det bra att göra det?&quot;</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Funktionsdelar</td>
<td>Kunna kommentera era funktioner på ett tydligt och motiverat sätt</td>
<td>A+B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A+B</td>
<td>Testers</td>
<td>&quot;Vad har ni skrivit för tester? Varför valde ni de ni valde?&quot;</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Funktionsdelar</td>
<td>Tillämpa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A+B</td>
<td>Beskriv algoritm med flödeschema</td>
<td>&quot;Har ni ritat ett flödeschema över er algoritm? Flödesdiagram ska finnas!&quot;</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Funktionsdelar</td>
<td>Analysera</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A+B</td>
<td>Funktionsdelar</td>
<td>&quot;Hur funkar er div-algoritm utan att referera till koden, så att någon som inte kan python kan förstå.&quot;</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Funktionsdelar</td>
<td>Analysera</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A+B</td>
<td>Funktionsdelar</td>
<td>&quot;Vad händer om jag anropar mul() och skickar in strängar istället?&quot;</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Funktionsdelar</td>
<td>Analysera</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A+B</td>
<td>Funktionsdelar</td>
<td>&quot;Hade vi kunnat använda en annan sorts loop för att skapa funktionerna? Hur hade det sett ut ungefär? Varför valde ni den loop ni valde?&quot;</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Funktionsdelar</td>
<td>Analysera</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A+B</td>
<td>Funktionsdelar</td>
<td>&quot;Vad kom ni fram till att de båda funktionerna hade för komplexitet?&quot;</td>
<td>mult(m,n) : O(n)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Funktionsdelar</td>
<td>Analys</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A+B</td>
<td>Funktionsdelar</td>
<td>&quot;Varför går mul snabbare än div med stora tal?&quot;</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Funktionsdelar</td>
<td>Bedöma</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A+B</td>
<td>Funktionsdelar</td>
<td>&quot;Jämför ni strängar eller heltal i era if-satser?&quot;</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Funktionsdelar</td>
<td>Diskussion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Student</td>
<td>Lärandemål</td>
<td>Moment / Fråga</td>
<td>Förväntat svar</td>
<td>Kommentar</td>
<td>Rätt</td>
<td>Lite fel</td>
<td>Fel</td>
<td>Underlag saknas</td>
<td>Bloom-nivåer</td>
<td>Info till Studenterna</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>----------------</td>
<td>----------------</td>
<td>-----------</td>
<td>------</td>
<td>---------</td>
<td>-----</td>
<td>---------------</td>
<td>-------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>A+B</td>
<td>Skapa hashtabeller</td>
<td>&quot;Vad får vi tillbaka om vi lägger in ett element med nyckel x och sedan söker på nyckel x?&quot;</td>
<td>Vi ska få tillbaka samma element som vi läter in</td>
<td>-</td>
<td>-</td>
<td>Skapa</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A+B</td>
<td>Skapa hashtabeller</td>
<td>Testfall: Lägg in strängen &quot;Hej!&quot; med nyckeln &quot;meddelande&quot;. Sök sedan efter nyckeln &quot;meddelande&quot;</td>
<td>Vi ska få tillbaka &quot;Hej!&quot;</td>
<td>De måste ha en fungerande hashtabell!</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A</td>
<td>Förklara datastruktur</td>
<td>&quot;Varför gör det snabbare att hitta element i en hashtabell jämfört med en vanlig lista?&quot;</td>
<td>Hashfunktionens körs bara en gång oavsett listans längd</td>
<td>Bedöma</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>Beräkna tidskomplexiteten för att leta upp ett element i en hashtabell om vi bortser från krockhanteringen?</td>
<td>O(1)</td>
<td>Diskutera gärna hur krockhanteringen kan påverka. Vad händar om allt krockar och hamnar i samma krocklista t.ex?</td>
<td>Analysera</td>
<td>Se ovan</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A</td>
<td>Skapa hashfunktioner</td>
<td>&quot;Vad bör en bra hashfunktion ha för egenskaper och varför?&quot;</td>
<td>Returnerar index i rätt intervall. Samma nyckel ska alltid ge samma hashvärde. Bra spredning, relativt snabb.</td>
<td>Förstå</td>
<td>Skapa en egen hashfunktion</td>
<td>Se ovan</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>Förklara algoritmer</td>
<td>&quot;Hur fungerar hashfunktionen ni skapat?&quot;</td>
<td>Skai i princip har egenskaperna från frågan ovan</td>
<td>De måste ha skapat en egen hashfunktion</td>
<td>Skapa</td>
<td>Se ovan</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A</td>
<td>Förklara krockhantering</td>
<td>&quot;Vad innebär en krock i en hashtabell?&quot;</td>
<td>Då två olika nycklar får samma hashvärde</td>
<td>Skapa</td>
<td>Förstå</td>
<td>Implementera och kunna förklara krockhantering med krocklistor eller probning</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>Implementera krockhantering / Förklara algoritmer</td>
<td>&quot;Vilken metod har ni använt för krockhantering och hur fungerar den?&quot;</td>
<td>Krocklistor eller någon sorts probning</td>
<td>De måste ha krockhantering</td>
<td>Skapa</td>
<td>Implementera och kunna förklara krockhantering med krocklistor eller probning</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A</td>
<td>Skapa hashtabell</td>
<td>&quot;Vilken storlek har ni valt på er tabell och varför?&quot;</td>
<td>Dubbla storleken bra tumregel, de ska ha försått att det inte ska vara för trångt på tabellen</td>
<td>Bedöma</td>
<td>Hashtabellen ska vara lagom stor</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>Skissa datastruktur</td>
<td>&quot;Skissa en bild av er tabell och vad som händer när ett element läggs till. Hur ser det ut vid en krock?&quot;</td>
<td>-</td>
<td>Förstå</td>
<td>Kunna skissa hashtabellen</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A+B</td>
<td>Förklara datastruktur</td>
<td>&quot;Med tanke på att samma artist förekommer flera gånger i listan av låtar, hur påverkar det hashtabellen om vi lägger in objekt med artistnamn som nyckel kontra låt-id som nyckel?&quot;</td>
<td>Vi kommer få färre poster i det första fallet, när samma artist dyker upp igen så skrivs den första låten över. Det kan håndha att dess implementation fungerar amorunda. Om de inte explicit skriver över den gamla låten, utan sparar flera med samma nyckel, finns det då något sätt att komma åt fler än den första som man stöter på vid sökning i tabellen?</td>
<td>Diskussionsfråga</td>
<td>Analysera</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Appendix B

Iteration 2
<table>
<thead>
<tr>
<th>Moment / Fråga</th>
<th>Förväntat svar</th>
<th>Kommentar</th>
<th>Rätt</th>
<th>Fel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skapa hashtabeller</td>
<td>Vad storlek har ni valt på er tabell och varför?</td>
<td>Dubbla storleken bra tumregel, de ska ha förtsått att det inte ska vara för trångt i af.</td>
<td>OK</td>
<td>Inte OK, om det är ett mindre fel så återkom senare under passet</td>
</tr>
<tr>
<td>Skissa datastruktur</td>
<td>Skissa en bild av er tabell och vad som händer när ett element läggs till.</td>
<td>Hur ser det ut vid en krock?</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>Skapa datastruktur</td>
<td>Med tanke på att samma artist förekommer flera gånger i listan av låtar, hur påverkar det hashtabellen om vi lägger in objekt med artistnamn som nyckel kontra låt-id som nyckel?</td>
<td>Vi kommer få färre poster i det första fallet, när samma artist dyker upp igen så skrivs den förra låten över. Det kan hända att deras implementation fungerar annorlunda. Om de inte explicit skriver över den gamla låten, utan sparar flera med samma nyckel, finns det då något sätt att komma åt fler än den första som man stöter på vid sökning i tabellen?</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>Förklara krockhantering</td>
<td>Vilken metod har ni använt för krockhantering och hur fungerar den?</td>
<td>Krocklistor eller någon sorts probning</td>
<td>A+B</td>
<td></td>
</tr>
<tr>
<td>Förklara krockhantering</td>
<td>Hur fungerar hashfunktionen ni skapat?</td>
<td>Ska i princip ha egenskaperna från frågan ovan - De måste ha skapat en egen hashfunktion</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>Förklara skapa hashtabeller</td>
<td>Vilken storlek har ni valt på er tabell och varför?</td>
<td>Dubbla storleken bra tumregel, de ska ha förtsått att det inte ska vara för trångt i af.</td>
<td>OK</td>
<td>Inte OK, om det är ett mindre fel så återkom senare under passet</td>
</tr>
<tr>
<td>Förklara skissa datastruktur</td>
<td>Skissa en bild av er tabell och vad som händer när ett element läggs till.</td>
<td>Hur ser det ut vid en krock?</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>Förklara skapa datastruktur</td>
<td>Med tanke på att samma artist förekommer flera gånger i listan av låtar, hur påverkar det hashtabellen om vi lägger in objekt med artistnamn som nyckel kontra låt-id som nyckel?</td>
<td>Vi kommer få färre poster i det första fallet, när samma artist dyker upp igen så skrivs den förra låten över. Det kan hända att deras implementation fungerar annorlunda. Om de inte explicit skriver över den gamla låten, utan sparar flera med samma nyckel, finns det då något sätt att komma åt fler än den första som man stöter på vid sökning i tabellen?</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>Förklara krockhantering</td>
<td>Vilken metod har ni använt för krockhantering och hur fungerar den?</td>
<td>Krocklistor eller någon sorts probning</td>
<td>A+B</td>
<td></td>
</tr>
<tr>
<td>Förklara skapa hashtabeller</td>
<td>Vilken storlek har ni valt på er tabell och varför?</td>
<td>Dubbla storleken bra tumregel, de ska ha förtsått att det inte ska vara för trångt i af.</td>
<td>OK</td>
<td>Inte OK, om det är ett mindre fel så återkom senare under passet</td>
</tr>
<tr>
<td>Förklara skissa datastruktur</td>
<td>Skissa en bild av er tabell och vad som händer när ett element läggs till.</td>
<td>Hur ser det ut vid en krock?</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>Förklara skapa datastruktur</td>
<td>Med tanke på att samma artist förekommer flera gånger i listan av låtar, hur påverkar det hashtabellen om vi lägger in objekt med artistnamn som nyckel kontra låt-id som nyckel?</td>
<td>Vi kommer få färre poster i det första fallet, när samma artist dyker upp igen så skrivs den förra låten över. Det kan hända att deras implementation fungerar annorlunda. Om de inte explicit skriver över den gamla låten, utan sparar flera med samma nyckel, finns det då något sätt att komma åt fler än den första som man stöter på vid sökning i tabellen?</td>
<td>OK</td>
<td></td>
</tr>
</tbody>
</table>
### Lärandemål

<table>
<thead>
<tr>
<th>Moment / Fråga</th>
<th>Förväntat svar</th>
<th>Kommentar</th>
<th>Rätt</th>
<th>Fel</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Utföra praktiska jämförelser av algoritmer</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fråga</strong></td>
<td><strong>Vad fick ni för resultat när ni tog tid på de olika delarna?</strong></td>
<td>(BE DEM VISA UPP SINA RESULTAT)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Förklaring av varje</strong></td>
<td>Linjärsökning: O(n)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Binärsökning: O(log n)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hashtabell-lookup: O(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Mätningar ska ha gjorts**

**Skapa A**

**Beräkna Tidskomplexitet**

**Fråga**

**Vad är komplexiteten för de tre olika sökningsalgoritmerna och varför blir de som de blir?**

**Förklara kortfattat varje**

- **Linjärsökning**: O(n)
- **Binärsökning**: O(log n)
- **Hashtabell-lookup**: O(1)

**Förklara**

**algoritm**

**Fråga**

**Vilken sorteringsmetod valde ni och hur fungerar den?**

Antingen Quicksort eller Mergesort, båda O(n log n)

De ska kunna förklara den valda metoden övergripande.

De ska inte bara använt pythons inbyggda, kod för sorteringen ska finnas.

**Kod för sorteringen (som de kan förklara hyfsat)** måste finnas!

**Analysera**

Visa hur dina sök-, och sorteringsfunktioner fungerar

**Utföra praktiska jämförelser av algoritmer**

**Fråga**

**Hur lade ni upp era tester, för att resultaten skulle bli tydliga och tillförlitliga?**

De ska ha kört med flera olika stora datamängder så at vi kan se hur de olika funktionernas körtid varierar.

**Tester måste ha gjorts!**

**Motivera upplägget av dina experiment**

**Utföra praktiska jämförelser av algoritmer**

**Fråga**

**Vad används parametern number till i timeit och varför är den bra att ha?**

Bestämmer antal körningar. Med många körningar kan vi få ett snittvärde, då körtiden kan variera beroende på andra processer som är i gang

**Förstå**

Läs om modulen timeit och besvara frågor om den

**Utföra teoretiska jämförelser av algoritmer**

**Fråga**

**I del 2, där ni ska plocka ut den n:te längsta låten, vid vilket värde på n kom ni fram till att sorteringen borde löna sig?**

Med 1 miljon låtar börjar det löna sig runt 20:e längsta. 1000000*log2(1000000) = ca 20 miljoner. Alltså lika mycket som 20 iterationer utan sortering.

Liknande resonemang som detta iaf.

De kan ha testat med en mindre del av låtlistan, det är ok, och de får då ett annat resultat. Men de ska ha resonerat rätt.

**Många missar det teoretiska resonemanget här, de ska ha skrivit ner en uträkning och ha en siffra som svar!**

**Analysera**

Se uppgift 2 i labyledningen

**Utföra praktiska jämförelser av algoritmer**

**Fråga**

**I del 2, vad fick ni för resultat när ni testade? Vad tror ni att eventuella skillnader kan bero på?**

Borde vara ungefär samma, men verkar kunna variera en del beroende på olika detaljer. Samma storleksordning borde det vara iaf.

Om det skiljer mycket här kan man diskutera lite hur det kan komma sig.

**De måste ha kört praktiska tester!**

**Skriv kortfattat om teorin stämmer med dina testresultat. Förklara kortfattat skillnaden i tid mellan de olika momenten.**

**OK**

Inte OK, om det är ett mindre fel så återkom senare under passet

Ej applicerbart

---

**Student**
Appendix C

Original lab descriptions in Swedish

The hash table lab, that appears in both of the courses appears here twice, since there are some small differences in how it is described. The descriptions are in this order:

- Lab 1
- Lab 2 (DD1321 version)
- Lab 3
- Lab 2 (DD1320 version)
Multiplikation och division

Table of Contents

- 1. lab p2
  - 1.1. Förberedelser
    - 1.1.1. github
  - 1.2. Programmeringsuppgift
    - 1.2.1. Multiplikation
    - 1.2.2. Division
    - 1.2.3. Hur multiplikation görs på riktigt
    - 1.2.4. Felhantering, division med noll
    - 1.2.5. Dokumentera funktionerna
    - 1.2.6. Testprogram
    - 1.2.7. Importera dina funktioner till ett testprogram
  - 1.3. Skriftlig algoritmbeskrivning
  - 1.4. Analys av algoritmen

1 lab p2

1.1 Förberedelser

1.1.1 github

Senare i kursen kommer vi att lära oss använda versionshanteringssystemet github. För att kunna generera kurskataloger åt var och en så behöver ni logga in på KTH:s github med ert KTH:login.

1.2 Programmeringsuppgift

I denna uppgift ska du implementera multiplikation och division av heltal enbart med hjälp av plus och minus.

1.2.1 Multiplikation

Skriv en funktion mult som tar två heltal (x, y) som parametrar och returnerar heltalen multiplicerat med varandra. Du får enbart använda dig av plus och minus. Den enkla lösningen är att du gör en loop som adderar x med sig självt under y varv.

Funktionen ska kunna multiplicera ett stort tal med ett litet tal utan att det tar orimligt lång tid. Prova med 12231231232312323 * 5 och 15 * 323241244132443.

1.2.2 Division

Skriv en funktion div som tar två heltal som parametrar (t, n) och returnerar t heltalsdividerat med n. Den enkla lösningen är att loopa x varv och i varje varv kolla om mult(n, x) är större än t för i så fall är x-1 svaret men det tar orimligt långt tid för stora tal.

Ditt programmet ska klara av stora tal utan att det tar orimlig tid och kom ihåg att du får enbart använda dig av plus, minus och din egen funktion mult.

Ett lite effektivare alternativ är att dela upp täljaren i deltal som divideras, ungefär som liggande stolen. Delten kan vara olika långa.

Använd strängfunktionalitet för att ta ut deltalen. Exempel:

```python
tal = 135212125
antalsiffror = len(str(tal))
treforstasiffror = int(str(tal)[0:3])
restenavsfifforna = int(str(tal)[3:antalsiffror])
```

1.2.3 Hur multiplikation görs på riktigt

En dator räknar med binära tal och algoritmerna för multiplikation och division utnyttjar hårdvaran (processorn) programmet körs på. Om man vill lära sig med om sådan programmering kan man läsa vidare i kurser som maskinära programmering. Syftet med denna labb är att lära sig algoritmämmande, strängmanipulering och testning.

1.2.4 Felhantering, division med noll

Om nämnaren är noll så ska du kasta ett undantagsfel ZeroDivisionError.

undantagsfel ZeroDivisionError
Mål: Kunna använda och implementera en hashtabell. Kunna använda unittest för att testa program.

Läs i Miller-Ranum om Hashing, och repetera Föreläsning 7

1. Hashning med Pythons inbyggda dictionary

Nu ska du göra en egen hashtabell med Pythons dictionary!

Skriv en klass DictHash som använder Pythons inbyggda dictionary. Den måste ha metoderna

- **store**(nyckel, data) som lagrar data som value i din dictionary, med nyckel som key.
- **x = search**(nyckel) som slår upp nyckel i din dictionary.

Dessutom får du om du vill lägga till två extra metoder:

- Vill du kunna skriva **d[nyckel]** istället för **d.search(nyckel)**? Definiera då metoden **__getitem__**(self, nyckel) som anropar din search-metod.
- Vill du kunna skriva **if nyckel in d**? Definiera då metoden **__contains__**(self, nyckel) som returnerar True om nyckel finns i d, False annars.

Testkör din klass DictHash med datafilen unique_tracks.txt. Du får själv välja vad du vill ha som nyckel, t ex artist eller låtnamn. Tips: split()

2. En egen implementation av hashtabellen

Nu ska du även göra hashningen själv, i din nya klass Hashtabl med samma gränssnitt och funktionalitet som DictHash.

Krav:

2. Hashtabellen ska vara lagom stor.
3. Du måste skriva en egen hashfunktion.
4. Någon krockhantering måste ingå, t ex krocklistor eller probning.
5. Du ska använda KeyError för att tala om att en nyckel inte finns.

Testning

- Prova med datafilen unique_tracks.txt (om det går långsamt, testa först att bara hasha de första 100 raderna i filen).
- Prova med en för liten hashtabell och kolla att krockhanteringen fungerar.

Redovisning

Vid redovisningen ska du kunna

- skissa hashtabellen,
- förklara varför hashning ger snabb sökning,

Läs i Miller-Ranum om Sorting and Searching.

Sökning och sortering

I denna labb ska du arbeta med större datamängder. Labben är uppdelad i delar. Läs igenom hela labben först.

Del 1

Filen unique_tracks.txt (84MB) är hämtad från Million Song Dataset. Den innehåller data om en miljon låtar. Varje rad i filen har formatet:

trackid<SEP>låtid<SEP>artistnamn<SEP>låttitel

Lista med objekt

Skriv en klass som representerar en låt enligt ovan. Förutom de vanliga metoderna ska du också implementera __lt__(self, other) som kan jämföra om objektet self är mindre än objektet other, med avseende på artistnamn.

Läs in låtarna från filen, skapa ett objekt för varje rad och spara objektken både

- i en lista (vektor)
- i en dictionary (med t.ex artistnamn som nyckel)

Testa att inläsningen har fungerat.

Modulen timeit

Läs i dokumentationen för Pythons modul timeit och svara på följande frågor:

- Vad representerar parametern stmt?
- Vad representerar parametern number?
- Vad är det timeit tar tid på?
- Vad skrivs ut av ett anrop av timeit?

Tidtagning

Nu ska du skriva ett program som gör följande, och tar tid på varje del:

- Söker efter ett element med linjärsökning i osorterade listor.
- Sorterar listor med mergesort eller quicksort som du kopierar (och förstår) från bok, nät eller föreläsningsanteckning (använd inte pythons inbyggda sorteringsmetod).
- Söker med binärsökning i sorterade listor.
- Slår upp element i dictionary

Som hjälp för att komma igång har du följande exempel (kursiverade funktioner behöver du skriva själv). När man tar tid på en funktion som har parametrar får man använda lambda. Ditt eget progam kan se annorlunda ut.

```python
def main():
    filename = "/info/tilda/unique_tracks.txt"
    # file_del2 = "/info/tilda/sang-artist-data.txt"
    lista, dictionary = readfile(filename)
    antal_element = len(lista)
    print("Antal element =", antal_element)
    sista = lista[n-1]
    testartist = sista.artist
    linjtid = timeit.timeit(stmt = lambda: linsok(lista, testartist), number = 1000)
    print("Linjärsökningsen tog", round(linjtid, 4) , "sekunder")
```

Du får gärna använda kod (för sortering och sökning) från föreläsningar, kursboken eller andra källor, men var noga med att ange var koden kommer från. Du får ändra och utföra egna tester där du sorterar på låttiteln istället för artistnamn, eller gör nya objekt där man t ex sorterar i första hand med avseende på artist och i andra hand med avseende på låttitel för att få en större sökmängd.

Du behöver köra flera körningar med olika testmängder (t.ex 1000, 10000, 50000, 200000). Du kan antingen tillverka flera testfiler, t.ex. genom att på kommandorad i unix/mac skriva

```
$ head -1000 /info/tilda/unique_tracks.txt > nyfilmed1000forstarader.txt
```

Läs in låtarna från filen, skapa ett objekt för varje rad och spara objektken både

- i en lista (vektor)
- i en dictionary (med t.ex artistnamn som nyckel)

Testa att inläsningen har fungerat.
Mål: Kunna använda och implementera en hashtabell, samt använda unittest för att testa ett program. Se Föreläsning 5, Övning 4

Läs i Miller-Ranum om Hashing.

1. Hashning med Pythons inbyggda dictionary

Minns du hur du i labb 2 gjorde en enkel kö med Pythons array? Nu ska du göra en egen hashtabell med Pythons dictionary!

Skriv en klass DictHash som använder dictionary. Den ska ha metoderna

- store(nyckel, data)
- x = search(nyckel)

Prova din dictionary med datafilen från förra labben. Tar det lika lång tid att köra?

Vill du kunna skriva d[nyckel] istället för d.search(nyckel)? Definiera då metoden __getitem__(self, nyckel)

Om du vill kunna skriva if nyckel in d definierar du metoden __contains__ (self, nyckel)

2. En egen implementation av hashtabellen

Nu ska du även göra hashningen själv och inte använda pythons dictionary internt. Din nya klass Hashtabell ska ha samma gränssnitt och funktionalitet som DictHash. Krav:

- Hashtabellen ska vara lagom stor
- Du måste skriva en egen hashfunktion, som ger en bra fördelning över hela tabellen
- Redogör för hur bra fördelningen är. T.ex. med en teoretisk analys av din algoritm eller genom att mäta hur många krockar det är som mest vid insättning.
- Någon krockhantering måste ingå, t ex krocklistor eller probning
- Du ska använda KeyError för att tala om att en nyckel inte finns

Testkörning 1

Prova med datafilen från förra labben. Du behöver inte använda enbart artist som nyckel.

Testkörning 2


Betyg

Denna labb kan endast ge betyg E. Du måste lämna in den och redovisa den i tid för att få göra labbarna för högre betyg i period 2.

Redovisning

Vid redovisningen ska du kunna

- motivera ditt val av hashfunktion, krockhantering och tabellstorlek,
- skissa hashtabellen,
- förklara varför hashning ger snabb sökning