

**Lab 1**

* **Computer algorithm Development**

Name: Rasmus Muhrbeck

Date: 2020-09-17

**A lab report in the course DT555A Programming in C**

Table of Contents

[1. Introduction 1](#_Toc51231182)

[2. Design 2](#_Toc51231183)

[3. Implementation and Test 5](#_Toc51231184)

[4. Results and discussion 6](#_Toc51231185)

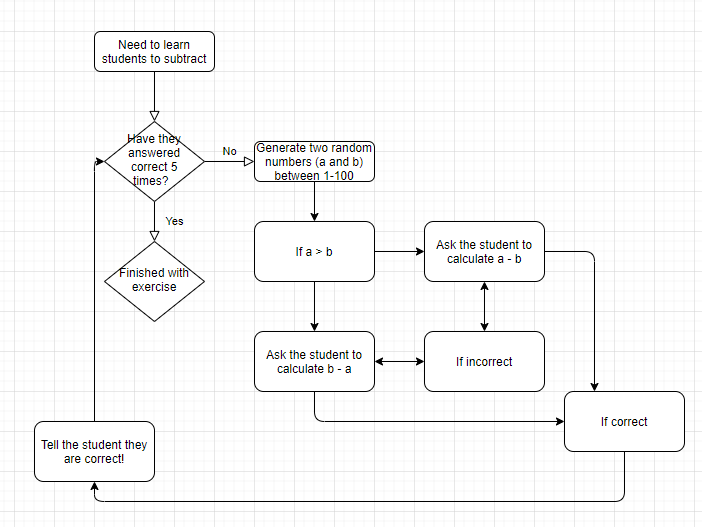
# 1. Introduction

This lab was made to practice algorithm development and representation in flowcharts and pseudo code and not in actual code. This way, the though process on how to solve coding problems can be simplified due to “ignoring” the complexity of coding. This is a great way to get an overview in text of how the problem should be solved.

# 2. Design

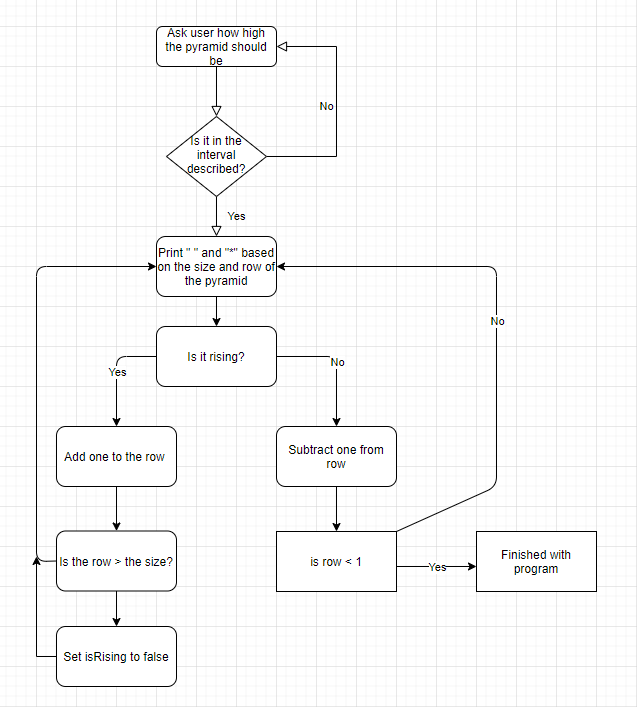
**Task 1 Grade 5: Computer Assisted Instructions**

When solving this problem my first thought was to have a counter that counts the amount of correct answers given by the student, and when it reached a certain number (5 in flowchart) it would be finished. While the student hasn’t answered enough correct guesses there would be 2 random numbers generated, lets call them a and b. Due to the fact that my primary school students don’t know negative numbers, I have to check if random number a is bigger than random number b. If that is the case, the program will ask the student to calculate a – b, and in the other case what b – a is. If the calculation is incorrect, the program will state that it was wrong and go back to asking the student to calculate the same a – b, or b – a, depending on the case. If the calculation is correct, the program will congratulate the student and after that check if the student has answered correct a certain amount of times. If the student still has some answers left, the program will state that the student only has answered correct X amount of times. The program will then ask the student to calculate another two random numbers a and b. This will continue until the correct amount of answers required is achieved, which then will end the program.



**Task 2 (Option 2) Grade 5: Pyramid**

My thought process here was to have a Boolean isRising which stated if the pyramid was on the rise, or on the fall. This helped a lot with the logic on how to print the pyramid. The program first asks the user to state how high the pyramid should be (between 5-20) and if the user states something outside that perimeter, the user will be asked again until inside the interval. Then the program should check if the pyramid is rising, if it is it should print “ “ for sizeOfPyramid – rowOfPyramid and “\*” for the remainder, which is just rowOfPyramid. It should then add one to the rowOfPyramid and check if the row is larger than the size. This will increase the amount of “\*” and decrease the amount of “ “. In the case that it is larger, it will swap so that the isRising is false (falling). In this loop it will instead of add one to rowOfPyramid, it will subtract one. When the situation where isRising is false and rowOfPyramid = 0, the program will be finished.



# 3. Implementation and Test

# 4. Results and discussion

The two pseudocodes that I described above have been tested and are working as intended. I like the idea of first writing a pseudo-code to simplify the program and break it down into smaller problems. This is a great way to get an overview of how one needs to structure the program to solve the task. By doing that the problems got quite easy to solve.