Högskolan Kristianstad
291 88 Kristianstad
044-20 30 00
www.hkr.se

DT555B **Programming in C**

Lab 2 – Algorithm Implementation

Objectives and Assessment

This lab is to provide you with practices on algorithm implementation in C and program testing. The objectives of this lab include the selection of right data types, implementation of control flow, and using standard input/output for reading data into the computer and printing the output for the user.

Implementation

This lab continues the work you did in lab1, by implementing the algorithms you developed. The grading policy is the same as in lab1:

It should be noted that not all tasks are required: you choose grade 3 tasks or grade 5 tasks. One lab report is required for each lab. You can extend your lab1 report by adding sections on algorithm implementation and program testing.

This lab is mandatory and individual. The discussion with your classmates are promoted. But the copy-and-paste is not allowed.

- You should find time yourself to implement your algorithms and write a lab report.
- You need to upload the report in due date so that the report is graded within 2 weeks after the due date.

Grade 3-- Task 1 Computer Assisted Instruction (CAI)

In this task, you are asked to implement the algorithm developed in lab1. To make it easier for you to read about this CAI task, the following text is taken from lab1.

The use of computers in education is referred to as computer assisted instruction (CAI). Write an algorithm that will help a primary school student practice additions (with numbers in the range 0—100). Your algorithm will generate an addition question and ask the student to give the answer. For example, your algorithm would print a question like

How much is 6 plus 7? Answer =

The student then enters the answer. Your algorithm checks the student's answer. If it is correct, the algorithm prints "very good!" and the algorithm stops. If the answer is wrong, it prints "No. Please try again." And then let the student try the same question again repeatedly until the student finally gets it right. Then the algorithm stops.

The key to this CAI algorithm is how to generate a question. We suppose that you can use the following pseudo-code to set a random number to a variable var:

var = get a random number in the range 0—100

So to generate an addition question, and get the answer from the student, we can have the following pseudo-code:

```
a = get a random number in the range 0—100
b = get a random number in the range 0—100
OUT "How much is the sum of a and b:  "
IN svar
```

Here, we use 2 variables a and b to keep 2 random values for the addition, while the answer is read into the variable named svar.

The steps to complete this task:

1) If you have not complete the algorithm development for this task, you should develop the algorithm and represent it in both flowchart and pseudo-code first.
2) Implement your algorithm in C.
3) Test your program to see if it works as desired. If not, you should repeat the process (4 steps about problem solving given in the lecture) until a correct solution in C is developed and tested.
4) Add implementation and testing to the lab report.

Grade 3-- Task 2 Multiplication Table

In this task, you are asked to implement the algorithm developed in lab1. To make it easier for you to read about this CAI task, the following text is taken from lab1.

Develop an algorithm for printing an n x n multiplication table of size n (n>0 and n<= 11). An example output is shown below, where the size of the table is 10 (input). Remember that you need to output the border of the table as shown in the example:

```
Enter the size of the table:  10
  x |    1    2    3    4    5    6    7    8    9   10
----------------------------------------------------------
  1 |    1    2    3    4    5    6    7    8    9   10
  2 |    2    4    6    8   10   12   14   16   18   20
  3 |    3    6    9   12   15   18   21   24   27   30
  4 |    4    8   12   16   20   24   28   32   36   40
  5 |    5   10   15   20   25   30   35   40   45   50
  6 |    6   12   18   24   30   36   42   48   54   60
  7 |    7   14   21   28   35   42   49   56   63   70
  8 |    8   16   24   32   40   48   56   64   72   80
  9 |    9   18   27   36   45   54   63   72   81   90
 10 |   10   20   30   40   50   60   70   80   90  100
```

Note: at the algorithm development, you do not need to consider how to produce the nicely formatted output. The formatted output will be realized later in lab, after the formatted output in C is discussed in Unit 3 Basic C.

The steps to complete this task:

1) If you have not complete the algorithm development for this task, you should develop the algorithm and represent it in both flowchart and pseudo-code first.
2) Implement your algorithm in C
3) Apply the test cases you developed in lab1 to check if your solution is correct. If not, you should repeat the process (4 steps about problem solving given in the lecture) until a correct solution in C is developed and tested.
4) Add implementation and testing to the lab report.

Grade 5-- Computer Assisted Instruction (CAI)

In this task, you are asked to implement the algorithm developed in lab1. To make it easier for you to read about this CAI task, the following text is taken from lab1.

The use of computers in education is referred to as computer assisted instruction (CAI). Write an algorithm that will help a primary school student practice subtractions (with numbers in the range 0—100). Your algorithm will generate a subtraction question and ask the student to give the answer. For example, your algorithm would print a question like

How much is 8 minus 6? Answer =

The student then enters the answer. Your algorithm checks the student's answer. If it is correct, the algorithm prints "very good!" and the algorithm stops. If the answer is wrong, it prints "No. Please try again." And then let the student try the same question again repeatedly until the student finally gets it right. The algorithm will stop when the student has completed **10** subtraction questions correctly.

The key to this CAI algorithm is how to generate a question. We suppose that you can use the following pseudo-code to set a random number to a variable var:

var = get a random number in the range 0—100

So to generate an addition question, and get the answer from the student, we can have the following pseudo-code:

```
a = get a random number in the range 0—100
b = get a random number in the range 0—100
OUT "How much is the a - b:  "
IN svar
```

Here, we use 2 variables a and b to keep 2 random values for the addition, while the answer is read into the variable named svar.

Since the primary school student has not learnt the negative numbers, your subtraction question should not have a negative answer. For example, your algorithm should not generate a subtraction like "6 – 8".

Please spend time to think how you could solve it. You need to consider the difference between selection and repetition structures, and nested loops to formulate your solution. When you are clear with the solution logic, you represent your solution in both pseudo-code and flowchart. You need to test it to see if it works correctly.

The steps to complete this task:

1) If you have not complete the algorithm development for this task, you should develop the algorithm and represent it in both flowchart and pseudo-code first.
2) Implement your algorithm in C
3) Test your program to see if it works as desired. If not, you should repeat the process (4 steps about problem solving given in the lecture) until a correct solution in C is developed and tested.
4) Add implementation and testing to the lab report

Grade 5 task 2 has 2 options. You can choose one to solve:

Grade 5-- Task 2 Computation of e^x

In this task, you are asked to implement the algorithm developed in lab1. To make it easier for you to read about this CAI task, the following text is taken from lab1.

This task requires knowledge on absolute value and factorial. You can read about factorial at https://en.wikipedia.org/wiki/Factorial.

The exponential function e^x can be represented by the Taylor series:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ..= \sum_{n=0}^{\infty} \frac{x^n}{n!},$$

where n! denotes the factorial of n. This Taylor series is an infinite series and it is not possible for you to implement an infinite loop to compute the function. In practice, it is enough to add the first **n terms** as a good approximation. The more terms are summed, the result is more accurate.

The requirements on the solution are

1) You can only use basic addition, subtraction, multiplication, and division in your solution. No math function like pow( ) should be used in the computation.
2) The value of e^x is obtained by summing up terms in the Taylor series until it finds that the absolute value of a term is less than 0.0000001.
3) Your solution should print out a table of values summing up 1, 2, 3, …n terms. A sample output is given below.

The steps to complete this task:

1) If you have not complete the algorithm development for this task, you should develop the algorithm and represent it in both flowchart and pseudo-code first.
2) Implement your algorithm in C
3) Apply the test cases you developed in lab1 to check if your solution is correct. If not, you should repeat the process (4 steps about problem solving given in the lecture) until a correct solution in C is developed and tested.
4) Add implementation and testing to the lab report.

```
Enter the value of x: 1
        terms   value
        1       1.0000000
        2       2.0000000
        3       2.5000000
        4       2.6666667
        5       2.7083333
        6       2.7166667
        7       2.7180556
        8       2.7182540
        9       2.7182788
        10      2.7182815
        11      2.7182818
```

Grade 5-- Task 2 Pyramid

In lab1, you have developed an algorithm to print a pyramid in the following form (example size is 5). The size should be limited to be between 5 and 20. To print the pyramid, you can only print one char at a time.

A sample pyramid of size 5 is given below:

```
    *
   **
  ***
 ****
*****
 ****
  ***
   **
    *
```

The steps to complete this task:

1) If you have not complete the algorithm development for this task, you should develop the algorithm and represent it in both flowchart and pseudo-code first.
2) Implement your algorithm in C
3) Apply the test cases you developed in lab1 to check if your solution is correct. If not, you should repeat the process (4 steps about problem solving given in the lecture) until a correct solution in C is developed and tested.
4) Add implementation and testing to the lab report.