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

DT555A **Programming in C**

Lab 2 – Computations

Objectives and Assessment

This lab is to provide you with practices on algorithm implementation in C. The objectives of this lab include the selection of right data types, implementation of control flow, and using standard input/output. You will also practice on program testing.

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. A report should also be submitted.

For grade 3: do Task 1 and Task 2. Grade 4 can be assigned if you complete it well and have written a good report.

For grade 5: do Task 3 and Task 4, and write a good report. The grade may be set to be 3 or 4, based on your completion and lab report. It should be noted that Tasks 3 and 4 have 2 options, you can choose any option, based on your interests and background.

Implementation

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 and develop algorithms to solve the problems, represent your algorithms in flowcharts and pseudo-codes, and write a lab report.

You need to upload the report in due date

Task 1 Multiplication Table

Develop a C program for printing a well formatted multiplication table. The input is the size of the table. An example output is shown below, when the desired size is 10 (input)::

```
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
```

In Lab1, you have developed your algorithm represented in flowchart/pseudo-code, and test cases. In this lab, you should

1) Implement your algorithm in C
2) 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.

Task 2 Computer Assisted Instruction (CAI)

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? 6 + 7 =

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

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 a:

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 = random number in the range 0—100
b = random number in the range 0—100
OUT "How much is ", a, " + ", b, " = "
IN svar
```

Here, we use 2 variables a and b to keep 2 random values for addition, while the answer is read into the variable named svar. In the output operation, we print the addition question (the text string given within the double quotes, and values of variables a and b.)

In lab1, you have developed an algorithm represented in flowchart/pseudo-code, and the test cases. In this lab, you should

1) Implement your algorithm in C
2) Test the 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.

Task 3 has 2 options. You can choose option 1 or 2.

Task 3 is required for grade 5.

Task 3 (Option 1) Computation of a Polynomial Using Horner's Method

Design and develop an algorithm for computing the polynomial

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0,$$

for a given value of x and the coefficients a_n, a_{n-1}, ..., a_1, a_0, using the Horner's method. Only additions and multiplications can be used in your solution (so you cannot use any pow( ) function to compute the x^i). You should not use an array in your solution either.

In Lab1, you have developed your algorithm represented in flowchart/pseudo-code, and test cases. In this lab, you should

1) Implement your algorithm in C
2) 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.

Task 3 (option 2) Computer Assisted Instruction (CAI)

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? 6 + 7 =

The student then types the answer. Your algorithm checks the student's answer. If it is correct, print "very good!" and then ask another addition question. If the answer is wrong, print "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 addition 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 a:

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 = random number in the range 0—100
b = 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 addition, while the answer is read into the variable named svar.

1) Implement your algorithm in C
2) Test the program to see if it works. 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.

Task 4 has 2 options. You can choose one of them.

Task 4 is required for grad 5.

Task 4 (option 1) Computation of e^x

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 added, 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 the absolute value of a term 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 as below.

```
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
```

In Lab1, you have developed your algorithm represented in flowchart/pseudo-code, and the test cases. In this lab, you should

1) Implement your algorithm in C
2) 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.

Task 4 (Option 2) Pyramid

Develop an algorithm to produce a pyramid in the following form (example size is 5). The size should be limited to be between 5 and 20.

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

In Lab1, you have developed your algorithm represented in flowchart/pseudo-code, and the test cases. In this lab, you should

1) Implement your algorithm in C
2) 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.