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

Lab4: Structured Program Development -- Computer-Assisted Instruction

Objectives and Assessment

This lab is to provide you further practice on algorithm development, implementation and program testing by solving a more complicated problem. The objectives of this lab include applying the top-down design to develop a solution to a complicated problem, implementation solutions to sub-problems in functions, and using arrays and strings.

Implementation

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

You should start to work on this lab before the scheduled lab 4 session, otherwise, you will not be able to complete it in due time.

You may be asked for individual presentation, if you do not carry out your lab in the scheduled lab session.

Task – Development and Implementation of a Simple CAI Program

The use of computers in education is referred to as computer-assisted instruction (CAI). The National Council of Teachers of Mathematics (NCTM) of United States of America, believed that "Technology is an essential tool for learning mathematics in the 21st century, and all schools must ensure that all their students have access to technology". A lot of research also showed that CAI had positive effects on students' achievement in mathematics. If you are interested, you can find more evidence on Internet. You may also read a doctoral thesis "The Effectiveness of Computer-Aided Instruction on Math Fact Fluency" at http://scholarworks.waldenu.edu/cgi/viewcontent.cgi?article=1028&context=dissertations (Links to an external site.)

In this lab task, you are to develop a simple CAI program for primary school students (so the numbers involved in the program are in the range 0…100). The program should contain at least the **following features** and satisfy the following requirements:

1) The user could choose to do **practices**, **test** or **exit** the program (main menu).

2) For a **practice**

The user should be able to choose doing the practices on additions only, subtractions only, or mixed additions with subtractions. The number of questions in a practice is fixed to be **10**. For each question, the user is asked to give the answer. If the answer is not correct, the program should continue to ask the user to give an answer, until a correct one is given. When the practice is completed, it should be back to the main menu to allow the user to choose practice, test or quit.

An example scenario is given as follows. The text in red is the input from the user, while all others are the output from the program.

Enter your name: eric

Wellcome, Eric!

You can choose:

1. do a practice
2. complete a test
3. quit the program

Enter your choice: 1

Now, you can choose to do practices on:

1. additions
2. subtractions
3. additions and subtractions

Enter your choice: 1

Now, you will be given 10 questions to solve:

1. 6 + 7 =

Enter your answer: 13

Very good!

2. 12 + 23 =

Enter your answer: 33

No. Please try again. Enter your answer: 35

Very good!

….

3) For a **test**

The user should be able to choose to do a test on additions only, subtractions only, or mixed additions with subtractions. The number of questions in the test is fixed to be **15**. When the test is completed, it should show the test result. It should be back to the main menu to allow the user to choose practice, test or quit.

When a test is completed, not only the test result is shown (in percentage), but also the questions, correct answers and user's answers are displayed. A sample output is given below:

```
Your test result is 86 (percentage)
Detailed questions and answers:

Nr      Question        Correct answer  Your answer
1       94 + 22         116             116
2       76 + 80         156             156
3       53 + 62         115             115
4       55 - 9          46              46
5       87 - 10         77              77
6       81 - 18         63              80
7       83 - 67         16              16
8       73 + 21         94              94
9       96 - 68         28              28
10      2 + 81          83              83
11      38 + 20         58              58
12      78 - 29         49              1
13      28 + 80         108             108
14      10 + 20         30              30
15      97 - 44         53              53
```

4) **Responses** to the answers in a practice

This is applied to the responses in doing practices. One problem that develops in CAI environments is user fatigue. This can be reduced by **varying** the computers dialog to hold the user's attention. Design the comments printed for each correct answer and each incorrect answer as follows:

Responses to correct answer:

Very good!
Excellent!
Nice work!
Well done!

Great!
Keep up the good work!

Responses to an incorrect answer:

No. Please try again.
Wrong. Try once again.
Don’t give up!
No. Keep trying.

5) Requirement on **subtraction** questions

Suppose that the user does not have the knowledge of negative numbers. So when you generate a random subtraction question, you should make sure the result is not negative. For example, you should not generate a question like

32 – 51

since the result is negative.

Structured program development and report

You should use the top-down design approach to divide the problem into smaller sub-problems, until you know how to solve the sub-problems. The solution to a sub-problem can be implemented as a function. You should not start the coding and implementation before you have done the design (and algorithms to solve all the sub-problem. You should try to have small functions, instead of some large functions in the program.

The report should include the top-down design process and the algorithm development/representation in flowchart or pseudo-code, before you discuss the implementation and program test.