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

DT555B **Programming in C**

Lab 3 Calendar/ CAI

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.

There are grade 3 and grade 5 tasks. You choose one that you are interested in lab 3. A lab report is required. You may get grade 3, 5, or 4, depending on which task you choose and how well it is solved, and report is written.

- You should find time yourself and develop algorithm(s) to solve problem(s), implement your solution, test your solution, and write a lab report.
- You need to upload the report in due date if you want the report is graded in time.

Grade 3 task has 2 options for you to choose.

Grade 3 Task Option 1-- A Simple Swedish Calendar

The Gregorian calendar, also called the Western calendar and the Christian calendar, is the most widely used calendar. It is supposed that the first January in year 0001, is Monday.

It should be mentioned: To avoid confusion about the calendar, you should not google too much and read about the calendars. The more you read, the more confusion you may get, since the calendar systems have been changed many time in history. What you need to know for this lab is that the first day in year 0001 is Monday. You use this as the reference date to calculate the week day for any day later on.

A simplified calendar format is shown in Format 1 at the end of this instruction. Your task is to develop a solution to print the calendar of a given year (the program should read in the year).

A core sub-problem in calendar task is to find the day of week for a given date. For example, the day of week for 2015.10.02 is Friday. You can use 0, 1, 2, 3, 4, 5, 6, to represent Sunday, Monday, Tuesday, Wednesday, Thursday, Friday and Saturday. In this lab, you ARE NOT ALLOWED to use any formula found on Internet to compute the day of week. You should use the following method:

1) Use the date 0001.01.01 as the reference date and that day is Monday.
2) If you want to find the day of week for date 2015.10.02, you just need to count the total days from 0001.01.01 through 2015.10.02
3) Day of week can be found by operation **total_days % 7** (remainder operator).

You should try some dates to make sure that you understand this method, before you start to implement this method to find the day of week. This is often the good way to understand how it works. For examples, let us check the days of week for the beginning days in year 0001:

Date:	1.1.1	1.1.2	1.1.6	1.1.7	1.1.8	1.1.9
Total days from 0001.01.01:	1	2	6	7	8	9
Day of week:	Mon(1)	Tue(2)	Sat(6)	Sun(0)	Mon(1)	Tus(2)

Here the total days from 0001.01.1 are inclusive. For example, from 0001.01.01 through 0001.01.08, the total days are 8. Then the day of week for 0001.01.08 is 8 % 7 = 1, that is, Monday. With these concrete examples, we can derive the calculation of day of week as follows:

Day of week = total days % 7

Where the operator % is integer remainder operator. For example, 8 % 7= 1 (remainder when do the division by 7), so 0001.01.08 is Monday. Also, remember that you need to consider if a year is a leap year or not. (You find it in 2.5 and 5.7 in English textbook, and in the example code **is-leap-year.c** in Module 4.

Some steps/issues that you need to consider when you develop your solution:

1) Understand the problem. What is the input and what is expected output?
2) Apply top-down design to divide the problem into some sub-problems and develop algorithms for them. It is good to have the pseudo-code/flowcharts before you start the implementation. When you have multiple sub-problems, implement each by a function, and the flowchart representation of the algorithm is one flowchart per function, one pseudo-code per function
3) Combine the solutions to print the calendar. A typical calendar output can be found at the end of the page.
4) Which test cases you need to test your solution? You need to test it to be sure that the printed calendar is correct.

Grade 3 Task Option 2-- A Simple CAI Program

This task continues the CAI program presented in lab1/2. 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

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) for the so it contains at least the following features:

1) The user could choose to do **practices**, **test** or quit 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 given the answer. If the answer is not correct, the program should continue 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.
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 (in percentage). 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
Welcome, Eric!
You can choose:
    1. do practices
    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!
 ….
```

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-problems). Otherwise, you may take even more time to manage this task.

There are 2 options for grade 5 task:

Grade 5 Task Option 1-- A Unix Calendar

On Unix (and Linux), you can use the program *cal* to print a calendar for a given year. The format is shown in Format 2 at the end of this instruction. Your task is to print the calendar in this format. As mentioned above, you are not allowed to use some formula found on the Internet to compute the day of week.

Read more about calendar in section Grade 3 Task Option 1 above.

In Sweden, we also number weeks. It would also be good to add week numbers to the calendar. But inclusion of week numbers is not required in this task.

If you like, you can add more features to your calendar, and make it as a Christmas present to your friends.

Some steps/issues that you need to consider when you develop your solution:

1) Understand the problem. What is the input and what is expected output?
2) Apply the top-down design to divide the problem into some sub-problems and develop algorithms for them. It is good to have the pseudo-code/flowcharts before you start the implementation
3) Implement the solutions to the sub-problems in functions, and test them
4) Combine the solutions to print the calendar
5) Which test cases you need to test your solution? You need to test it to be sure that the printed calendar is correct.

Grade 5 Task Option 2 -- A Computer Aided Instruction Program

This Task is further extension to CAI tasks in lab1/2. The main instruction for this task is the same as Grade 3 Task Option 2 above. But you should have the extra requirements as given below:

1. Responses to the answers
   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.

2. Show the results of a test
   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. For example:
   Your test result is 80 (percentage)
   Detailed questions and answers:

Question	correct answer	your answer
1. 6 + 7	13	13
2. 12 + 23	35	34
3. ……		

3. 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.

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-problems). Otherwise, you may take even more time to manage this task.

Format 1: A common calendar format in Sweden (Task 1 Option 1)

Enetr year: 2015

2015

January	February	March	April	May	Jur
1 Th	1 Su	1 Su	1 We	1 Fr	1 Mo
2 Fr	2 Mo	2 Mo	2 Th	2 Sa	2 Tu
3 Sa	3 Tu	3 Tu	3 Fr	3 Su	3 We
4 Su	4 We	4 We	4 Sa	4 Mo	4 Th
5 Mo	5 Th	5 Th	5 Su	5 Tu	5 Fr
6 Tu	6 Fr	6 Fr	6 Mo	6 We	6 Sa
7 We	7 Sa	7 Sa	7 Tu	7 Th	7 Su
8 Th	8 Su	8 Su	8 We	8 Fr	8 Mo
9 Fr	9 Mo	9 Mo	9 Th	9 Sa	9 Tu
10 Sa	10 Tu	10 Tu	10 Fr	10 Su	10 We
11 Su	11 We	11 We	11 Sa	11 Mo	11 Th
12 Mo	12 Th	12 Th	12 Su	12 Tu	12 Fr
13 Tu	13 Fr	13 Fr	13 Mo	13 We	13 Sa
14 We	14 Sa	14 Sa	14 Tu	14 Th	14 Su
15 Th	15 Su	15 Su	15 We	15 Fr	15 Mo
16 Fr	16 Mo	16 Mo	16 Th	16 Sa	16 Tu
17 Sa	17 Tu	17 Tu	17 Fr	17 Su	17 We
18 Su	18 We	18 We	18 Sa	18 Mo	18 Th
19 Mo	19 Th	19 Th	19 Su	19 Tu	19 Fr
20 Tu	20 Fr	20 Fr	20 Mo	20 We	20 Sa
21 We	21 Sa	21 Sa	21 Tu	21 Th	21 Su
22 Th	22 Su	22 Su	22 We	22 Fr	22 Mo
23 Fr	23 Mo	23 Mo	23 Th	23 Sa	23 Tu
24 Sa	24 Tu	24 Tu	24 Fr	24 Su	24 We
25 Su	25 We	25 We	25 Sa	25 Mo	25 Th
26 Mo	26 Th	26 Th	26 Su	26 Tu	26 Fr
27 Tu	27 Fr	27 Fr	27 Mo	27 We	27 Sa
28 We	28 Sa	28 Sa	28 Tu	28 Th	28 Su
29 Th		29 Su	29 We	29 Fr	29 Mo
30 Fr		30 Mo	30 Th	30 Sa	30 Tu
31 Sa		31 Tu		31 Su	

July	August	September	October	November	December
1 We	1 Sa	1 Tu	1 Th	1 Su	1 Tu
2 Th	2 Su	2 We	2 Fr	2 Mo	2 We
3 Fr	3 Mo	3 Th	3 Sa	3 Tu	3 Th
4 Sa	4 Tu	4 Fr	4 Su	4 We	4 Fr
5 Su	5 We	5 Sa	5 Mo	5 Th	5 Sa
6 Mo	6 Th	6 Su	6 Tu	6 Fr	6 Su
7 Tu	7 Fr	7 Mo	7 We	7 Sa	7 Mo
8 We	8 Sa	8 Tu	8 Th	8 Su	8 Tu
9 Th	9 Su	9 We	9 Fr	9 Mo	9 We
10 Fr	10 Mo	10 Th	10 Sa	10 Tu	10 Th
11 Sa	11 Tu	11 Fr	11 Su	11 We	11 Fr
12 Su	12 We	12 Sa	12 Mo	12 Th	12 Sa
13 Mo	13 Th	13 Su	13 Tu	13 Fr	13 Su
14 Tu	14 Fr	14 Mo	14 We	14 Sa	14 Mo
15 We	15 Sa	15 Tu	15 Th	15 Su	15 Tu
16 Th	16 Su	16 We	16 Fr	16 Mo	16 We
17 Fr	17 Mo	17 Th	17 Sa	17 Tu	17 Th
18 Sa	18 Tu	18 Fr	18 Su	18 We	18 Fr
19 Su	19 We	19 Sa	19 Mo	19 Th	19 Sa
20 Mo	20 Th	20 Su	20 Tu	20 Fr	20 Su
21 Tu	21 Fr	21 Mo	21 We	21 Sa	21 Mo
22 We	22 Sa	22 Tu	22 Th	22 Su	22 Tu
23 Th	23 Su	23 We	23 Fr	23 Mo	23 We
24 Fr	24 Mo	24 Th	24 Sa	24 Tu	24 Th
25 Sa	25 Tu	25 Fr	25 Su	25 We	25 Fr
26 Su	26 We	26 Sa	26 Mo	26 Th	26 Sa
27 Mo	27 Th	27 Su	27 Tu	27 Fr	27 Su
28 Tu	28 Fr	28 Mo	28 We	28 Sa	28 Mo
29 We	29 Sa	29 Tu	29 Th	29 Su	29 Tu
30 Th	30 Su	30 We	30 Fr	30 Mo	30 We
31 Fr	31 Mo		31 Sa		31 Th

Format 2: Linux/Unix calendar format (Task 2 Option 1)

```
Enetr year: 2015
                                2015

      January               February               March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                 1  2  3   1  2  3  4  5  6  7   1  2  3  4  5  6  7
 4  5  6  7  8  9 10   8  9 10 11 12 13 14   8  9 10 11 12 13 14
11 12 13 14 15 16 17  15 16 17 18 19 20 21  15 16 17 18 19 20 21
18 19 20 21 22 23 24  22 23 24 25 26 27 28  22 23 24 25 26 27 28
25 26 27 28 29 30 31                       29 30 31

       April                  May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                 1  2  3  4                       1  2  3  4  5  6
 5  6  7  8  9 10 11   3  4  5  6  7  8  9   7  8  9 10 11 12 13
12 13 14 15 16 17 18  10 11 12 13 14 15 16  14 15 16 17 18 19 20
19 20 21 22 23 24 25  17 18 19 20 21 22 23  21 22 23 24 25 26 27
26 27 28 29 30        24 25 26 27 28 29 30  28 29 30
                      31

       July                  August                September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                 1  2  3  4                          1  2  3  4  5
 5  6  7  8  9 10 11   2  3  4  5  6  7  8   6  7  8  9 10 11 12
12 13 14 15 16 17 18   9 10 11 12 13 14 15  13 14 15 16 17 18 19
19 20 21 22 23 24 25  16 17 18 19 20 21 22  20 21 22 23 24 25 26
26 27 28 29 30 31     23 24 25 26 27 28 29  27 28 29 30
                      30 31

      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                 1  2  3   1  2  3  4  5  6  7          1  2  3  4  5
 4  5  6  7  8  9 10   8  9 10 11 12 13 14   6  7  8  9 10 11 12
11 12 13 14 15 16 17  15 16 17 18 19 20 21  13 14 15 16 17 18 19
18 19 20 21 22 23 24  22 23 24 25 26 27 28  20 21 22 23 24 25 26
25 26 27 28 29 30 31  29 30                 27 28 29 30 31
```