CSC / CIS 175
Problem Solving and Programming - I

University of Michigan-Flint
Department of Computer Science, Engineering, and Physics (CSEP)

COLLEGE OF ARTS & SCIENCES
COMPUTER SCIENCE, ENGINEERING, & PHYSICS

Fall 2013

November 10, 2013

Homework 9

(100 points)

due by November 20, Wednesday 8:00am

Remarks:

- No emailed homeworks will be accepted.
- Only submission is via the BB system.
- No late submissions will be accepted.
- Individual assignment. No collaboration or group work is permitted.

Questions for the deliverable:

1. De Morgan's Law: De Morgan's law can sometimes make it more convenient for us to express a logical expression. These laws state that the following expressions are logically equivalent:

$$!\,(condition1\ \ \&\&\ \ condition2) \equiv (!\,condition1\ \ ||\ \ !\,condition2) \tag{1}$$

   Also,

$$!\,(condition1\ \ ||\ \ condition2) \equiv (!\,condition1\ \ \&\&\ \ !\,condition2) \tag{2}$$

   Use De Morgan's laws to write equivalent expressions for each of the following, then write a program with four functions (one for each proof) to show that the original expression and the new expression in each case are equivalent: (refer to logical operator example on p.199 of the slides)

   (a) !(x < 5) && !(y ≥ 7)

(b) !(a == b) || !(g != 5)

(c) !((x $\leq$ 8) && (y > 4))

(d) !((i > 4) || (j $\leq$ 6))

Expected Output:

```
PART A

!( x < 5 ): true
!( y >= 7 ): true
!(x < 5) && !(y >= 7) is equivalent to !((x < 5) || (y >= 7))

!( x < 5 ): true
!( y >= 7 ): false
!(x < 5) && !(y >= 7) is equivalent to !((x < 5) || (y >= 7))

!( x < 5 ): false
!( y >= 7 ): true
!(x < 5) && !(y >= 7) is equivalent to !((x < 5) || (y >= 7))

!( x < 5 ): false
!( y >= 7 ): false
!(x < 5) && !(y >= 7) is equivalent to !((x < 5) || (y >= 7))



PART B

!( a == b): true
!( g != 5): true
!(a == b) || !(g != 5) is equivalent to !((a == b) && (g != 5))

!( a == b): true
!( g != 5): false
!(a == b) || !(g != 5) is equivalent to !((a == b) && (g != 5))

!( a == b): false
!( g != 5): true
!(a == b) || !(g != 5) is equivalent to !((a == b) && (g != 5))

!( a == b): false
!( g != 5): false
!(a == b) || !(g != 5) is equivalent to !((a == b) && (g != 5))
```

```
PART C

( x <= 8 ): true
( y > 4 ): true
!((x <= 8) && (y > 4)) is equivalent to !(x <= 8) || !(y > 4)

( x <= 8 ): true
( y > 4 ): false
!((x <= 8) && (y > 4)) is equivalent to !(x <= 8) || !(y > 4)

( x <= 8 ): false
( y > 4 ): true
!((x <= 8) && (y > 4)) is equivalent to !(x <= 8) || !(y > 4)

( x <= 8 ): false
( y > 4 ): false
!((x <= 8) && (y > 4)) is equivalent to !(x <= 8) || !(y > 4)



PART D

( i > 4 ): true
( j <= 6 ): true
!((i > 4) || (j <= 6)) is equivalent to !(i > 4) && !(j <= 6)

( i > 4 ): true
( j <= 6 ): false
!((i > 4) || (j <= 6)) is equivalent to !(i > 4) && !(j <= 6)

( i > 4 ): false
( j <= 6 ): true
!((i > 4) || (j <= 6)) is equivalent to !(i > 4) && !(j <= 6)

( i > 4 ): false
( j <= 6 ): false
!((i > 4) || (j <= 6)) is equivalent to !(i > 4) && !(j <= 6)
```

2. An integer is said to be a perfect integer if the sum of its factors, including 1, but not the number itself, is equal to the number. For example, 6 is a perfect integer because 6 = 1 + 2 + 3. Write a function (perfect) that determines whether a given number is a perfect integer. Use this function in a program that determines and prints all perfect numbers between 1 and 1000. Print the factors of each perfect integer to confirm. Put your computer to hard work to find the fifth perfect integer. First is 6, second is 28, etc.

Expected Output:

```
 (i) for 1000 (under Windows DOS or Command prompt)

D:\temp\175-W07\hw9-ans>time /t & hw9-q2.exe & time /t
09:45 AM
Perfect integers between 1 and 1000:
6 = 1 + 2 + 3
28 = 1 + 2 + 4 + 7 + 14
496 = 1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248

09:45 AM


under Linux:
$> date; ./hw5-q5; date

(ii) run to find the fifth perfect integer
(answer not given here for obvious reasons)
```

3. Write a program that simulates the rolling of two dice. The program should use rand() to roll the first die and should use rand() again to roll the second die. The sum of the two values should then be calculated. Your program should roll the dice 120,000 times. Tally the numbers of times each possible sum appears. Print the results in tabular format. Also, determine if the totals are reasonable, i.e. there are six ways to roll a 7, so approximately one-sixth of all the rolls should be 7.

   Expected Output:

```
       Sum     Total  Expected   Actual
         2      3373    2.778%    2.811%
         3      6707    5.556%    5.589%
         4      9955    8.333%    8.296%
         5     13151   11.111%   10.959%
         6     16763   13.889%   13.969%
         7     19922   16.667%   16.602%
         8     16714   13.889%   13.928%
         9     13445   11.111%   11.204%
        10      9927    8.333%    8.273%
        11      6742    5.556%    5.618%
        12      3301    2.778%    2.751%
```

4. Write a program that helps an elementary school student learn multiplication. Use rand to produce two positive one-digit integers. It should then type a question such as

   How much is 6 times 7?

   The student types the answer. Your program checks the answer. If it is correct, print 'Very Good', then ask another multiplication question. If wrong, print 'No, please try again'. Let the student repeatedly try until the right answer.

   The various comments for correct and incorrect answers are printed as follows:

   Correct answer responses:
   - Very good!
   - Excellent!
   - Nice work!
   - Keep up the good work!

   Incorrect answer responses:
   - No. Please try again.
   - Wrong. Try once more.
   - Don't give up.
   - Not really, keep trying.

   Use random number generator to choose a number from 1 to 4 to select an appropriate response to each answer. Use switch statement to issue the responses.

   Expected Output:

```
Enter -1 to End.
How much is 0 times 7 (-1 to End)? 0
Nice work!

How much is 7 times 5 (-1 to End)? 35
Keep up the good work!

How much is 7 times 5 (-1 to End)? 35
Very good!

How much is 6 times 9 (-1 to End)? 54
Excellent!

How much is 8 times 8 (-1 to End)? 4
No. Keep trying.
? 5
No. Keep trying.
? 6
Don't give up!
```

```
? 7
No. Please try again.
? 3
No. Keep trying.
? 5
Wrong. Try once more.
? 6
No. Please try again.
? 7
No. Please try again.
? 8
Don’t give up!
? 6
Wrong. Try once more.
? 64
Nice work!

How much is 7 times 0 (-1 to End)? -1
That’s all for now. Bye.
```

5. Write a program using a recursive function to display the first 25 Fibonacci numbers.

   Expected Output:

```
Fibonacci (  1 ) =      0
Fibonacci (  2 ) =      1
Fibonacci (  3 ) =      1
Fibonacci (  4 ) =      2
Fibonacci (  5 ) =      3
Fibonacci (  6 ) =      5
Fibonacci (  7 ) =      8
Fibonacci (  8 ) =     13
Fibonacci (  9 ) =     21
Fibonacci ( 10 ) =     34
Fibonacci ( 11 ) =     55
Fibonacci ( 12 ) =     89
Fibonacci ( 13 ) =    144
Fibonacci ( 14 ) =    233
Fibonacci ( 15 ) =    377
Fibonacci ( 16 ) =    610
Fibonacci ( 17 ) =    987
Fibonacci ( 18 ) =   1597
Fibonacci ( 19 ) =   2584
Fibonacci ( 20 ) =   4181
Fibonacci ( 21 ) =   6765
Fibonacci ( 22 ) =  10946
Fibonacci ( 23 ) =  17711
Fibonacci ( 24 ) =  28657
Fibonacci ( 25 ) =  46368
```

6. `Extra credit (5%)`: A robot can take steps of 1 meter or 2 meters. Write a recursive program to calculate the number of ways the robot can walk n meters. Refer to the slides for more information.

   Expected Output:

   ```
   (i)
   Distance of the walk?  5
   The robot can walk 5 meters in 8 ways.

   (ii)
   Distance of the walk?  9
   The robot can walk 9 meters in 55 ways.

   (iii)
   Distance of the walk?  12
   The robot can walk 12 meters in 233 ways.
   ```

7. `Extra credit (5%)`: The tower of Hanoi is a puzzle consisting of three pegs mounted on a board and n disks of various sizes with holes in their centers, stacked from largest to smallest on the first peg. Your objective: to move all the rings to the second or third tower. You can only move the top ring on a given tower, and you can't place larger rings on top of smaller rings. Write a recursive program that solves this puzzle for the number of disks specified by the user. See the slides for more information.

Expected Output:

```
(i)
How many disks do you want to have?  3
Move top disk from peg O to D
Move top disk from peg O to S
Move top disk from peg D to S
Move top disk from peg O to D
Move top disk from peg S to O
Move top disk from peg S to D
Move top disk from peg O to D
```

```
(ii)
How many disks do you want to have?  5
Move top disk from peg O to D
Move top disk from peg O to S
Move top disk from peg D to S
Move top disk from peg O to D
Move top disk from peg S to O
Move top disk from peg S to D
Move top disk from peg O to D
Move top disk from peg O to S
Move top disk from peg D to S
Move top disk from peg D to O
Move top disk from peg S to O
Move top disk from peg D to S
Move top disk from peg O to D
Move top disk from peg O to S
Move top disk from peg D to S
Move top disk from peg O to D
Move top disk from peg S to O
Move top disk from peg S to D
Move top disk from peg O to D
Move top disk from peg S to O
Move top disk from peg D to S
Move top disk from peg D to O
Move top disk from peg S to O
Move top disk from peg S to D
Move top disk from peg O to D
Move top disk from peg O to S
Move top disk from peg D to S
Move top disk from peg O to D
Move top disk from peg S to O
Move top disk from peg S to D
Move top disk from peg O to D
```

Deliverables:

1. Source Code: (**.cpp** file) that must start with a comment block similar to the following:

```
/**************************************************************************
** Author          : Suleyman Uludag
** Program           : hw1, q1
** Date Created      : September 15, 2013
** Date Last Modified : September 16, 2013
** Usage             : No command line arguments
**
** Problem:
Accept the following information from the user (keyboard):
- Hw1, hw2 and hw3 (out of 100)
- Midterm (out of 100)
- Final exam (out of 100)
Calculate the total grade out of 100 based on the following grading scale:
Hws          -->    30% (10% each)
Midterm      -->    30%
Final Exam   -->    40%
** Pseudocode:
** 1)
** 2)
**************************************************************************/
```

2. Executable (.exe file under windows). You must explicitly state the platform of your executable (such as Linux, etc.) if it is not Windows. Please name your file by using the question number: **hw1-q1.exe** (for Windows)

3. Screenshot of your app. For screenshot, you can use the following free program on windows:

   `http://www.wisdom-soft.com/downloads/setupscreenhunterfree.exe`

   For Linux/Unix, there are many alternatives. I personally like shutter.
   File naming convention example:

   `hw1_q1.png` (or .jpg or another graphics format)

4. You must zip all the above three files into ONE .zip file and submit your assignment by the deadline on moodle system. Name your file as Lastname-Firstname-hw#.zip. For example, **Uludag-Suleyman-hw1.zip**

   For generating .zip file, you may use the following free software on Windows:

   `http://www.7-zip.org/download.html`

   Linux/Unix has many built-in.