Programming Assignment 1
(C++ fundamentals)
Due Date: September 13, 2013
Weightage: 10% of the final grade

Instructions:

- This assignment is to be turned in via Blackboard before **11:59 PM** on **September 13, 2013**. You should submit '.cpp' files with names mentioned after each problem.

- Make sure your programs compile (without any compiler errors). You will not receive any credit if your program does not compile. If you are unable to complete any of the programs, submit the parts that work (with no compiler errors) for partial credit.

- You can use any C++ compiler of your choice. If you do not have one, I recommend you to download the DevC++ compiler. It is simple and easy to use. If you use any other compiler or IDE (Integrated Developement Environment) like Eclipse or XCode, make sure that your program compiles as a stand alone C++ program. Submit ONLY one (.cpp) file for each problem. DO NOT submit a project folder or object/executable code). Your programs will be recompiled (using DevC++) for grading.

- Your grade will be based on functionality (does the program do what it is suppose to do), readability (is the code nicely formatted), and understandability (are the literals meaningful and is the code modular and well documented with appropriate comments). Try to incorporate all the good programming practices and styles. E.g., **WRITE COMMENTS**, declare all constants and variables at the top of a function, store quantities using the appropriate data types, declare function prototypes, etc.

---

1. Madlib (20 points)

Write a program that plays the game of Mad Lib. Your program should prompt the user to enter the following strings:

- The first or last name of your Instructor
- Your name
- A food
- A number between 100 and 120
- An adjective
- A color
- An animal

After getting the strings as input, they should be substituted into the story below. The resulting story should be displayed as the output.

Dear Professor [**Instructor Name**],

I am sorry that I am unable to turn in my homework at this time.
First, I ate a rotten [**Food**], which made me turn [**Color**] and
extremely ill. I came down with a fever of [**Number 100-120**].
Next, my [**Adjective**] pet [**Animal**] must have smelled the remains
of the [**Food**] on my homework, because he ate it. I am currently
rewriting my homework and hope you will accept it late.

Sincerely,
[**Your Name**]

Filename: madlib.cpp

---

2. Rock, Paper, Scissor (30 points)

Write a program to score the paper-rock-scissor game. Each of the two players type in either P, R, or S. The program then announces the winner based on the rule: *Paper covers Rock, Rock breaks Scissors, Scissors cut Paper, or Nobody wins.* Be sure to allow the players to use lowercase as well as uppercase letters. Your program should allow the users to play repeatedly until they say they are done. Your program should also keep a score for each player and display that after each play.

Sample run:

```
Player 1: Please enter either (R)ock, (P)aper, or (S)cissors: P
Player 2: Please enter either (R)ock, (P)aper, or (S)cissors: s

Player 2 wins.

Scores after this play:
    Player 1: 0
    Player 2: 1

Thanks!!

Play again? Y/y continues, other quits: n
```

Filename: rps.cpp

---

3. Pig game (50 points)

The game of Pig is a simple two player dice game in which the first player to reach 100 or more points wins. Players take turns. On each turn a player rolls a six-sided die. After each roll:

- If the player rolls a 2-6 then he can either:
  - ROLL AGAIN or
  - HOLD. At this point the sum of all rolls made this turn is added to the player's total score and it becomes the other player's turn.
- If the player rolls a 1 then the player loses his turn. He gets no new points and it becomes the opponent's turn.

If a player reaches 100 or more points after holding then the player wins.

Write a program that plays the game of Pig, where one player is a human and the other is the computer. Allow the human to input 'r' (or 'R') to roll again or 'h' (or 'H') to hold.

The computer program should play according to the following rule: Keep rolling on the computer's turn until it has accumulated 20 or more points, then hold. Of course, if the computer wins or rolls a 1 then the turn ends immediately. Allow the human to roll first.

Write your program using at least two functions:

int humanTurn(int humanTotalScore);
int computerTurn(int computerTotalScore);

These functions should perform the necessary logic to handle a single turn for either the computer or the human. The input parameter is the total score for the human or computer. The functions should return the turn total to be added to the total score upon completion of the turn. For example, if the human rolls a 3 and 6 and then holds, then humanTurn should return 9. However, if the human rolls a 3 and 6 and then a 1, then the function should return 0.

HINT: Note that the description of the humanTurn and computerTurn functions indicate that an outside variable must keep track of the total score by adding to itself the value returned by the functions at the completion of each player's turn. Also, to implement roll of a dice you can use the *srand(int)* and *rand()* functions from <cstdlib>. HINT: A random number between M and N (both inclusive, where M < N) is equal to *((rand() % (N-M+1)) + M)*.

Filename: pigGame.cpp

BONUS PROBLEM: Memory Matching (50 points)

A common memory matching game played by young children is to start with a deck of cards that contain identical pairs. For example, given six cards in the deck, two might be labeled '1', two might be labeled '2' and two might be labeled '3'. The cards are shuffled and placed face down on the table. The player then selects two cards that are face down, turns them face up, and if they match they are left face up. If the two cards do not match they are returned to their original position face down. The game continues in this fashion until all cards are face up.

Write a program that plays the memory matching game. Use sixteen cards that are laid out in a 4x4 square and are labeled with pairs of numbers from 1 to 8. Your program should allow the player to specify the cards that she would like to select through a coordinate system.

For example in the following layout

```
     1   2   3   4
   -------------------
1  |  8   *   *   *
2  |  *   *   *   *
3  |  *   8   *   *
4  |  *   *   *   *
```

all of the cards are face down except for the pair 8s, which has been located at coordinates (1,1) and (2,3). To hide the cards that have been temporarily placed face up, output a large number of newlines to force the old board off the screen.

Hint: Use a 2D array for the arrangement of cards and another 2D array that indicates if a card is face up or face down. Write a function that "shuffles" the cards in the array by repeatedly selecting two cards at random and swapping them. You can use the *srand(int)* and *rand()* functions from <cstdlib> to select cards at random. HINT: A random number between M and N (both inclusive, where M < N) is equal to *((rand() % (N-M+1)) + M)*.

Filename: memory.cpp