CSCI 1300 CS1: Starting Computing
Instructor: Fleming/Wong, Spring 2019
Project 1: Jeopardy Dice, 5% of your grade
Due Saturday, February 16th, by 6 pm
No bonus points for early submission

All 3 components (Cloud9 workspace, Moodle CodeRunner attempts, and zip file) must be completed and submitted by Saturday, February 16th, 6:00 pm for your homework to receive points. Project 1 requires you to have an interview grading with your TA by March 3rd.

---

1. Objectives

- Understand and work with `while` loops.
- Understand and work with `if-else` conditionals and `switch` statements.
- Writing and testing C++ functions
    - Understand problem description
    - Design your function:
        - come up with a step by step algorithm,
        - convert the algorithm to pseudocode
        - imagine many possible scenarios and corresponding sample runs or outputs
    - Convert the pseudocode into a program written in the C++ programming language
    - Test it in the Cloud9 IDE, then submit it for grading on Moodle

---

2. Background

While Loops

Loops allow us to run a section of code multiple times. They will repeat execution of a single statement or group of statements as long as a specified condition continues to be satisfied. If the condition is not true, then the statement will not be executed.

<u>Syntax and Form:</u>

```
while (condition)
{
 //statement(s) to do something;
}
```

Here *while* is a C++ reserved word, *condition* should be a Boolean expression that will evaluate to either **true** or **false**, and *statement to do something* is a set of instructions enclosed by curly brackets. If the condition is true, then the specified statement(s) within the loop are executed. After running once, the Boolean expression is then re-evaluated. If the condition is true, then the specified statement(s) are executed again. This process of evaluation and execution is repeated until the condition becomes false.

<u>Example 1:</u>

```
int userChoice = 1;
while (userChoice != 0)
{
   cout << “Do you want to see this question again? “ << endl;
   cout << “Press 0 for no, any other number for yes.” << endl;
   cin >> userChoice;
}
```

Entering ‘0’ will terminate the loop, but any other number will cause the loop to run again. Note how we must initialize the condition before the loop starts. Setting `userChoice` equal to 1 ensures that the while loop will run at least once.

<u>Example 2:</u>

```
int i = 0;
while (i < 5)
{
     cout << i << endl;
     i = i + 2;
}
```

Notice how you must manually initialize `i` to equal 0 and then manually increment `i` by 2. Inserting `print` statements into your loops is a quick way to debug your code if something isn't working, to make sure the loop is iterating over the values you want to be using. A common error is to forget to update `i` within the loop, causing it to run forever.

---

3. Submission Requirements

All three steps must be fully completed by the submission deadline for your project to be graded.

1. ***Create Project1 directory on your Cloud 9 workspace:*** Your recitation TA will review your code by going to your Cloud9 workspace. *TAs will check the last version that was saved before the submission deadline*.
   - Create a directory called **Project1** and place all your file(s) for this assignment in this directory.
   - Make sure to *save* the final version of your code (File > Save). Verify that this version displays correctly by going to File > File Version History.
   - The file(s) should have all of your functions, test cases for the functions in `main()` function(s), and adhere to the style guide. Please read the **submission file instructions** under week 4.

2. ***Submit to the Moodle CodeRunner:*** Head over to Moodle to the link **Project 1 CodeRunner**. You will find one programming quiz question for each problem in the assignment. Submit your solution for each problem and press the Check button. You will see a report on how your solution passed the tests, and the resulting score for the first problem. You can modify your code and re-submit (press *Check* again) as many times as you need to, up until the assignment due date.

3. ***Submit a .zip file to Moodle:*** After you have completed all 3 questions from the Moodle assignment, zip all 5 files you compiled in Cloud9 (one cpp file for each of three problems, plus two output text files), and submit the zip file through the **Project 1 (File Submission)** link on Moodle.

---

4. Rubric

Aside from the points received from the [Project 1 CodeRunner]() quiz problems, your TA will look at your solution files (zipped together) as submitted through the **Project 1 (File Submission)** link on Moodle and assign points for the following:

Comments **(5 points):**

- Your code should be well-commented. Use comments to explain what you are doing, especially if you have a complex section of code. These comments are intended to help other developers understand how your code works. These comments should begin with two backslashes (//) or the multi-line comments (/* … *comments here…* */) .
- Please also include a comment at the top of your solution with the following format:

```
// CS1300 Spring 2019
// Author: my name
// Recitation: 123 – Favorite TA
// Cloud9 Workspace Editor Link: https://ide.c9.io/…
// Project 1 - Problem # ...
```

Algorithm **(5 points)**

- Before each function that you define, you should include a comment that describes the inputs and outputs of your function and what algorithms you are using inside the function.
- This is an example C++ solution. Look at the code and the algorithm description for an example of what is expected.

Example 1:

```
/*
 * Algorithm: convert money from U.S. Dollars (USD) to Euros.
 *    1. Take the value of number of dollars involved in the transaction.
 *    2. Current value of 1 USD is equal to 0.86 euros
 *    3. Multiply the number of dollars got with the currency
 *       exchange rate to get Euros value
 *    4. Return the computed Euro value
 * Input parameters: Amount in USD (double)
 * Output (prints to screen): nothing
 * Returns: Amount in Euros (double)
 */
```

Example 2:

```
double convertUSDtoEuros(double dollars)
{
      double exchange_rate = 0.86; //declaration of exchange rate
      double euros = dollars * exchange_rate; //conversion
      return euros; //return the value in euros
}
```

The algorithm described below does not mention in detail what the algorithm does and does not mention what value the function returns. Also, the solution is not commented. This would work properly, but would not receive full credit due to the lack of documentation.

```
/*
 * conversion
 */
double convertUSDtoEuros(double dollars)
{
      double euros = dollars * 0.86;
      return euros;
}
```

Test Cases

In your Cloud9 solution file, for each function in the first two problems, you should have 2 test cases present in their respective `main()` function, for a total of 4 test cases. Your test cases should follow the guidelines, **Writing Test Cases**, posted on Moodle under Week 3.

Please make sure that your submission files follow the the **submission file instructions** under Week 5.

5. Start With These Problems...

… but be sure to have a look at #6 to plan your time, because it is a much larger task! These will prepare you for tackling #6, Jeopardy Dice.

Problem 1 (5 points): printOddNums

Write a function named **printOddNums** to print all positive even integers less than or equal to a max value.

- Your function **MUST** be named **printOddNums**
- Your function takes one input argument:
    - an **integer** parameter representing the max value.
- Your function should not return any value
- Your function should print the positive odd integers less than or equal to the max value (see expected output format below)

Example:

- When the input argument is 11, the expected output should be:

```
1
3
5
7
9
11
```

In Cloud9 the file should be called **printOddNums.cpp** and it will be one of 5 files you need to zip together for the **Project 1 (File Submission)** on Moodle.

Don't forget to head over to Moodle to the link **Project 1 CodeRunner**. For Problem 1, in the Answer Box, paste **only your function definition, not the entire program**. Press the Check button. You can modify your code and re-submit (press Check again) as many times as you need to.

Note: For problems 1 and 2, you can modify your code and re-submit (press Check again) up to 15 times without penalty. Additional attempts beyond 15 will incur an increasing penalty up to 50%.

Problem 2 (5 points): diceSum

Write a function **diceSum** which takes one integer argument for a desired sum. The function repeatedly rolls two six-sided dice until their sum is the desired sum.

- Your function **MUST** be named **diceSum**
- Your function takes one **integer** argument for the desired sum
- Your function should not return anything.
- If the desired sum is not between 2 and 12 (inclusive), then print the following message and does not roll the dice.

```
The desired sum should be between 2 and 12
```

- The function prints for each roll in the following output

```
You rolled a <die1> and a <die2>
```

- The function prints following message once we got the desired sum

```
Got a <desiredSum> in <number> rolls!
```

- Make sure that the outcome of the first roll is printed first, and the outcome of the second roll is printed second
- Please use the **rollDie** function given below to simulate the roll of a single die.

```
/* rollDie
 * returns a random integer between 1 and 6,
 * works as rolling a die.
 * return value, int number (1-6)
 */

int rollDie()
{
     return random() % 6 + 1;
}
```

In Cloud9 the file should be called **diceSum.cpp** and it will be one of 5 files you need to zip together for the **Project 1 (File Submission)** on Moodle.

Don't forget to head over to Moodle to the link **Project 1 CodeRunner**. For Problem 2, in the Answer Box, paste **only your function definition, not the entire program**. Press the Check button.
Note: For problems 1 and 2, you can modify your code and re-submit (press Check again) up to 15 times without penalty. Additional attempts beyond 15 will incur an increasing penalty up to 50%.

6. Jeopardy Dice

(Problem 3, 60 points)

This project's objective is to create a Jeopardy Dice game in which one player (the user) competes against the computer to reach 80 points. To start this project, download the template file **jeopardyDice.cpp** from Moodle.

Game Mechanics

- There are two players: the user and the computer, who alternate turns until one of them reaches 80 points or higher.
- The user is always the first player and starts the game.
- If either player reaches 80 points or more at the end of their turn, the game ends immediately and the other player does not get another turn.
- During one turn the player (either the user or computer) accumulates a total number of points equal to `turnTotal` by repeatedly rolling a single fair 6-sided die.
    - At the beginning of the player's turn, `turnTotal` starts at 0
    - If they roll a 2 or 5, their turn ends and the `turnTotal` is equal to 0.
    - If they roll a 4, their turn ends and `turnTotal` is equal to 15 (irrespective what they collected up to that roll)
    - If they roll a 1, 3, or 6, `turnTotal` increases by the amount on the die and the player rolls again
    - The value on the die is selected randomly during every roll
- At the end of the turn, a player's total score, `playerTotal` (either user or computer based on the turn), is increased by `turnTotal`.
- During a player's turn, **before** each roll, they are given two options:
    - Continue rolling
    - Hold; if the user chooses to Hold the user's turn ends, `turnTotal` is added to `playerTotal,` and the turn is passed to the computer.
- If the player is the computer, they will always continue rolling until their `turnTotal` reaches the value 10 or higher, at which point the computer player will end its turn and play passes to the user.

Specifications: You are given the following template (in the file **jeopardyDice.cpp**, posted on Moodle):

```
// CSCI1300 Spring 2019
// Author: myFirstName myLastName
// Recitation: 0123 - yourTAname
// Cloud9 Workspace Editor Link: <https://ide.c9.io/ .....>
// Project1
//Put your pseudocode for the entire game here (see template in
jeopardyDice.cpp)

#include <iostream>
#include <cmath>
#include <cstdlib>
#include <unistd.h>
#include <stdio.h>
using namespace std;

/**
 * rollDie
 * returns a random integer between 1 and 6, works as rolling a die.
 * return value, int number (1-6)
 */

int rollDie()
{
     return random() % 6 + 1;
}

// your 3 + functions go in here ...

/**
 * game ()
 * driver function to play the game
 * the function does not return any value
 */
void game()
{
     // your solution goes here
}
int main()
{
     // start the game!
     game();
     return 0;
}
```

- Your solution must have an algorithm in pseudocode explaining how you are approaching the problem, step by step.

- Your `main()` function will have just one statement: a call to the function `game()`. **Do not modify `main()`.**
- `game()` function is the driver function of your program and we will be testing your `game()` function on Moodle (CodeRunner).
  - The function does not take any input parameters.
  - The function does not return any value.
- Your solution should have at least 3 functions besides the `game()` and `rollDie()` functions. Since the functions you will create will be called from inside the `game()` function, their definition should be above the definition of `game()`.
- You are given the freedom to design the functions of your choice. But you must have a minimum of 3 functions of your choice.
- Creating functions that do nothing is not permitted. Each function should have a purpose and should be well-documented.
- Out of the three functions (or more) you will create …
  - one of the functions must have at least 1 input parameter,
  - one of the functions must return a value, and
  - one of the functions should not return anything, but it should print/display something.
- When CodeRunner grades your code, it expects that your solution does not have extra roll of dice. Everytime you call `rollDie()` function, be sure to use it.

Input and Output
You may assume that the user will only enter valid input. Notice that the program only prompts the user if they want to roll again if it is the user's turn, and that the prompt is case insensitive (the user may type either "y" or "Y", "n" or "N").

As the game progresses, we will use output statements to check the progress of the user (or the computer) during a turn, and at the end of each turn. Below are two possible examples of program output. User input is shown **in bold**. Your program should **reproduce** the output shown below (excluding differences due to random values).

Example outputs

Example Output 1	**Example Output 2**
Welcome to Jeopardy Dice! It is now human's turn Do you want to roll a dice (Y/N)?: **y** You rolled a 3	Welcome to Jeopardy Dice! It is now human's turn Do you want to roll a dice (Y/N)?: **n** computer: 0

```
Your turn total is 3
Do you want to roll again (Y/N)?:
y
You rolled a 1
Your turn total is 4
Do you want to roll again (Y/N)?:
y
You rolled a 6
Your turn total is 10
Do you want to roll again (Y/N)?:
y
You rolled a 3
Your turn total is 13
Do you want to roll again (Y/N)?:
n
computer: 0
human: 13

It is now computer's turn

Computer rolled a 5
Computer turn total is 0
computer: 0
human: 13

It is now human's turn

Do you want to roll a dice (Y/N)?:
Y
You rolled a 3
Your turn total is 3
Do you want to roll again (Y/N)?:
y
You rolled a 6
Your turn total is 9
Do you want to roll again (Y/N)?:
N
computer: 0
human: 22

It is now computer's turn

Computer rolled a 5
Computer turn total is 0
computer: 0
human: 22
```

```
human: 0

It is now computer's turn

Computer rolled a 1
Computer turn total is 1
Computer rolled a 3
Computer turn total is 4
Computer rolled a 6
Computer turn total is 10
computer: 10
human: 0

It is now human's turn

Do you want to roll a dice (Y/N)?:
n
computer: 10
human: 0

It is now computer's turn

Computer rolled a 6
Computer turn total is 6
Computer rolled a 5
Computer turn total is 0
computer: 10
human: 0

It is now human's turn

Do you want to roll a dice (Y/N)?:
n
computer: 10
human: 0

It is now computer's turn

Computer rolled a 1
Computer turn total is 1
Computer rolled a 3
Computer turn total is 4
Computer rolled a 2
Computer turn total is 0
computer: 10
human: 0
```

```
It is now human's turn

Do you want to roll a dice (Y/N)?:
y
You rolled a 1
Your turn total is 1
Do you want to roll again (Y/N)?:
y
You rolled a 6
Your turn total is 7
Do you want to roll again (Y/N)?:
y
You rolled a 4
Your turn total is 15
computer: 0
human: 37

[suppose output from many turns is here… ]

It is now human's turn

Do you want to roll a dice (Y/N)?:
y
You rolled a 2
Your turn total is 0
computer: 55
human: 71

It is now computer's turn

Computer rolled a 5
Computer turn total is 0
computer: 55
human: 71

It is now human's turn

Do you want to roll a dice (Y/N)?:
y
You rolled a 3
Your turn total is 3
Do you want to roll again (Y/N)?:
y
You rolled a 1
Your turn total is 4
Do you want to roll again (Y/N)?:
y
```

```
[omitted]

It is now computer's turn

Computer rolled a 3
Computer turn total is 3
Computer rolled a 3
Computer turn total is 6
Computer rolled a 4
Computer turn total is 15
computer: 81
human: 4

It is now human's turn

Do you want to roll a dice (Y/N)?:
n
computer: 61
human: 4

It is now computer's turn

Computer rolled a 4
Computer turn total is 15
computer: 76
human: 4

It is now human's turn

Do you want to roll a dice (Y/N)?:
n
computer: 76
human: 4

It is now computer's turn

Computer rolled a 4
Computer turn total is 15
computer: 91
human: 4

Congratulations! computer won this round of
Jeopardy Dice!
```

You rolled a 3 Your turn total is 7 Do you want to roll again (Y/N)?: **y** You rolled a 6 Your turn total is 13 Do you want to roll again (Y/N)?: **n** computer: 55 human: 84 Congratulations! human won this round of Jeopardy Dice!	

Testing
It is your responsibility to test your program. The random element of this program makes it so that you should not expect to exactly reproduce the example outputs. However, aside from the specific numbers, your output (the `cout` messages) should exactly match these examples.

7. Submitting files and Interview grading

Submitting the assignment:

1. Complete the **Project 1 CodeRunner**.
2. Create your output files. The output1.txt and output2.txt should be the output of your jeopardy game. What you need to do:
    a. Play the game (type y or n) until you or computer wins
    b. Copy the output on your screen and paste it to a txt file (name it output1.txt)
    c. Repeat one more time to create another file, output2.txt
    d. You can use `srand(123);` in your main to get different sequence of random numbers. Don't put srand() in the code runner. Otherwise, your solution may not pass the test cases.
3. Zip the following 5 files into one file, name it: firstName_lastName_project1.zip
    a. printOddNumber.cpp
    b. diceSum.cpp
    c. jeopardyDice.cpp

    d. **output1.txt**
    e. **output2.txt**
4. Submit the zip file to **Project 1 (File Submission)**
5. Sign up to the interview grading slot on Moodle. Please make sure that you sign-up and complete an interview grading with your TA by March 3rd. The schedulers for interview grading will be available after the deadline of this project.

   You are allowed to reschedule without any penalty only one time during the semester. If you miss a second time, you will be allowed to reschedule but you will incur a 25 points (out of 100) penalty, then 50 points for the third time.

8. Project 1 points summary

Criteria	**Pts**
CodeRunner	70
Comments	5
Algorithms	5
Interview grading	40
Recitation attendance (Feb 12 or Feb 14)*	-30
Total	120

* if your attendance is not recorded, you will lose points. Make sure your attendance is recorded on Moodle.