ITP 165: Introduction to C++ Programming

Homework 7: High-card simulator — Due: 11:59pm, March 11th, 2016

Goal

For this assignment, you'll be creating a simulator for an exciting game of High-Card. 2 players randomly pick 2 cards from two different decks. Whoever gets the highest rank (suit doesn't matter), wins the round. You'll run the simulator until the user asks for the excitement to stop.

Setup

- Create a new project in Visual Studio or Xcode called **hw07** and a new C++ file named **hw07.cpp**.
- Your **hw07.cpp** file must begin with the following (replace the name and email with your actual information):

```
// Name
// ITP 165, Spring 2016
// Homework 07
// USC email
```

Part 1: Struct setup

- Create a `struct` called `Card`. It has 2 pieces of data:
  - An `int` to hold the card's rank – a number from 2 to 14. Where 11 represents a Jack, 12 is a Queen, 13 is a King, and 14 is an Ace.
  - A `std::string` to hold the card's suit. Suits can be Diamonds, Clubs, Hearts, or Spades.
- Write `main` that, for now, makes a variable of type `Card`, just to test everything.

Part 2: cardToString function

- Create a function called "`cardToString`". It should accept 1 `Card` variable for input and return a `std::string`. This function will examine the inputted card and generate a string containing something like "2 of Clubs" or "Queen of Diamonds".
- I recommend creating a `std::string` to hold the return value.
- If the card rank is between 2 and 10, you can set the return value to contain the starting number like "5" or "10".
- If the card rank is between 11 and 14, you'll need to decode the rank to Jack, Queen, King, or Ace.
- Next, add the suit of the `Card` variable to the return value and return that new `std::string`.
- For now, inside of `main`, create a new `Card` variable for testing the `cardToString` function. Set the card's rank and suit to your favorite playing card value.
- You should be able to display out that card's information with code similar to:

```
std::cout << cardToString(testCard) << std::endl;
```

- Be sure to test your function with several `Card` values/variables.

Part 3: makeCard function

- Create a function called "`makeCard`". Its job is to create `Card` variables for us. It accepts 2 input:

    - An `int` to represent the rank of the final `Card` it will create.
    - A `std::string` that holds the suit of the final `Card` it will create.

  It returns a variable of type `Card`.
- Inside of `makeCard`, create a new `Card`, set the rank and suit of the new `Card` as the inputted values. Then return the new `Card`.
- Test the new function in `main`. Create a new `Card` with the new function. Your code should resemble:

```
Card myCard = makeCard(4, "Hearts");
```

- You should also try to display the information about the new `Card` to be sure things are working properly. Try this with a few `Card` variables.

Part 4: getRandRank and getRandSuit functions

- We'll be creating random numbers now, so we need to initialize a random number generator in `main`.
- Random number generation is accessible through the `cstdlib` library, so add a line at the beginning of your program to include that library
    - The random number generation system must be primed before you can create numbers. The function that primes the random number generator is `std::srand`. It returns no output, but receives an `int` as input – this is sometimes called the "seed". However, we need to choose the seed carefully. Because it initializes the random number generator, it shouldn't be the same each time we run our program.
    - A number that is always changing is time. In fact, the function `std::time` can return the ever-changing number of seconds since 1/1/1970. It just needs the input of zero.
- Prime the random number generation system in `main`. Make it the first line of code in `main`. To prime the system, call `std::srand`. It's input should be a call to `std::time` with 0 as its input, like this:

```
std::srand(std::time(0));
```

- The function that now can create a random number is `std::rand()`. It accepts no input and returns a random `int` as output. The random number will be somewhere between 0 and (about) 32000 (inclusive). Somehow you've got to reduce that random number's range for our purposes.
    - HINT: If you mod (%) the result, you restrict the upper bound exclusively. Meaning, a call such as `std::rand() % 6` would return a random number between [0, 6) - including 0 and excluding 6.
- Create a function called "`getRandRank`". It accepts no input and returns an `int` holding a random number between 2 and 14.
    - HINT: Think about how to get a random number between 0 and 12, then move the range up by 2.
- Create a function called "`getRandSuit`". It accepts no input and returns a `std::string` holding a random suit.
    - HINT: Generate a random number, and decide which suit corresponds to the result.
- Test the function in `main`. Create a new `Card` with the new functions. Your code should resemble:

```
Card myCard = makeCard(getRandRank(), getRandSuit());
```

- You should also try to display the information about the new `Card` to be sure things are working properly. If you run your program several times, you should display several, different `Card` values.

Part 5: Comparison functions

- Create 3 new functions:
  - The function "`isEqual`" will accept 2 `Card` variables as input and return a `bool`. If the 2 inputted `Card` variables have the same rank, `isEqual` returns `true`, otherwise it returns `false`.
  - The function "`isLess`" accepts 2 `Card` variables as input and return a `bool`. If the first inputted `Card` variable has a lower rank than the second inputted `Card` variable, `isLess` returns `true`, otherwise it returns `false`.
  - The function "`isGreater`" accepts 2 `Card` variables as input and return a `bool`. If the first inputted `Card` variable has a higher rank than the second inputted `Card` variable, `isGreater` returns `true`, otherwise it returns `false`.
- Test these new functions in `main`. Create 2 or 3 new `Card` variables, then compare them against each other them with `if` statements. Your code should resemble:

```
if (isGreater(myCard, yourCard))
{
        // myCard is higher!
}
```

  It might be a GoodIdea™ to display information about the `Card` variables too.

Part 6: The simulation

- Comment out (or delete) your test code from `main`. The only thing that should remain now is the random number initialization and the `return` statement.
- For the simulation, I recommend the following variables
  - For each player: store the player's name, their score, and their `Card`.
  - For the simulation: keep track of the round with a counter. Each time the user tells the program to run again, increment the round counter.
- Announce the beginning of your program, and who the match is between. If you'd like, you can ask the user for the 2 players' names – but you don't have to.
- Each round, generate a random card for the 2 players and determine whose card is higher.
  - The player with the higher card wins the round and you should increment their score. Make sure to inform the user who won the round.
- After the round is over, ask the user if the simulation should continue. Continue the simulation until the user prompts your program to quit.
- Output for Part 6 should look something like this (user input in **`red`**):

```
Welcome to the world champion High-card tournament finals!
It's a match between Larry and Curly!

They're starting round 1.
The players are shuffling...
```

```
Isn't this exciting!
Curly's 6 of Hearts beat Larry's 2 of Diamonds!
The score is...
Larry : 0, Curly : 1
Should the 2 players go again (y/n)? y

They're starting round 2.
The players are shuffling...
Look at that!
Larry's 8 of Hearts beat Curly's 7 of Hearts!
The score is...
Larry : 1, Curly : 1
Should the 2 players go again (y/n)? y

They're starting round 3.
The players are shuffling...
I didn't see that coming!
Larry drew a 4 of Diamonds to match Curly's 4 of Diamonds!
It's a DRAW!
The score is...
Larry : 1, Curly : 1
Should the 2 players go again (y/n)? n

Well that's all we have time for tonight.
For ESPN, this is Bob Jarvis, reporting live!
```

A Note on Style

Be sure to comment your code.

As we discussed in lecture, it is extremely important that your code is properly indented as it greatly adds to readability. Because of this, if you submit a code file that is not reasonably indented, you will have points deducted.

Likewise, you will lose points if your variable names are not meaningful. Make sure you use variable names that correspond to what you are actually storing in the variables.

Full Sample Output

Below is sample output for a full run-through of the program. Your output should resemble the following – but doesn't need to match it exactly (user input is in **red**).

```
Welcome to ESPN's coverage of the world champion High-card tournament finals!
It's a match between Larry and Curly!

They're starting round 1.
The players are shuffling...
Look at that!
Larry's 9 of Diamonds beat Curly's 4 of Hearts!
```

```
The score is...
Larry : 1, Curly : 0
Should the 2 players go again (y/n)? y

They're starting round 2.
The players are shuffling...
Isn't this exciting!
Curly's King of Hearts beat Larry's 9 of Hearts!
The score is...
Larry : 1, Curly : 1
Should the 2 players go again (y/n)? y

They're starting round 3.
The players are shuffling...
Look at that!
Larry's 9 of Diamonds beat Curly's 2 of Diamonds!
The score is...
Larry : 2, Curly : 1
Should the 2 players go again (y/n)? y

They're starting round 4.
The players are shuffling...
Look at that!
Larry's 8 of Diamonds beat Curly's 3 of Spades!
The score is...
Larry : 3, Curly : 1
Should the 2 players go again (y/n)? n

Well that's all we have time for tonight.
For ESPN, this is Bob Jarvis, reporting live!
```

Deliverables

A compressed folder containing **hw07.cpp**. This can be done by:

a. WINDOWS:
   i. Select your file
   ii. Right click
   iii. Send to ->
   iv. Compressed (zipped) folder
   v. Make sure this compressed folder is named **hw07.zip**
   vi. Submit this compressed folder through Blackboard

b. OSX:
   i. Select your file
   ii. Right click
   iii. Compress **"hw07.cpp"**
   iv. Make sure this compressed folder is named **hw07.cpp.zip**
   v. Submit this compressed folder through Blackboard

Grading

Item	Points
Part 1: Struct setup	5
Part 2: cardToString function	5
Part 3: makeCard function	5
Part 4: getRandRank and getRandSuit functions	10
Part 5: Comparison functions	5
Part 6: The simulation	10
Total*	**40**

* Points will be deducted for poor code style, or improper submission.