CS216: Introduction to Software Engineering Techniques (Spring, 2018)
Project Assignment 1
(100 points)
Today's Date: Monday, February 12
Due Date: Monday, February 26

(You should include your name and section number as part of the comments for your project assignment!!!)

Problem Statement

Write a program that allows the user to play a *simplified* poker game, named Texas Holdem (https://en.wikipedia.org/wiki/Texas_hold_%27em), against the computer.
Two cards, known as the hole cards, are dealt face down to each of the user and the computer respectively, and then five community cards (also called "shared cards" or "window cards") are dealt face up in the center of the table and shared by both players. Each player then combines the hole cards with the community cards (2 cards + 5 cards) to seek the best five-card poker hand from any combination of the seven cards. Your program should then compare the highest ranking of the five-card poker hands between the user and the computer, then report the one, who has higher ranking of the five-card poker hand, wins the game, or otherwise report a tie game. (If you are not familiar with card point values and suits of cards, please refer to the description in Lab4 document)

The following is the rule to decide the highest ranking of the five-card poker hands, from higher to lower ranking (please note that the rule is a subset of wikipedia's ***Hand-ranking categories*** (https://en.wikipedia.org/wiki/List_of_poker_hands):

- **Straight flush** (5 cards are of sequential card value (points), and all of the same suit)
- **Four of a kind** (4 cards are of the same card value and 1 card of another card value, suit has no impact)
- **Full House** (3 cards are of the same card value, and 2 cards are of another same card value, suit has no impact)
- **Flush** (5 cards are of the same suit, not all of the sequential card value)
- **Straight** (5 cards are of sequential card value, not all of the same suit)
- **Three of a kind** (3 cards are of the same card value, and 2 cards are of two other different card points, suit has no impact)
- **Pair** (any 2 cards are of the same card value, suit has no impact)
- **High Card** (simply nothing above)

Besides the hand-ranking categories above, if two five-card poker hands are of the same hand-ranking category, your program then also need to check the highest card value (points) of each category to finally decide the higher ranking: if two five-card poker hands are of the same hand-ranking category, the higher card value beats the other. ***The card value of each hand-ranking category*** is defined as follows:

- **Straight flush** (card value is the highest card points of the five sequential cards)
- **Four of a kind** (card value is the card points of the same four cards)

- **Full House** (card value is the card points of the same three cards)
- **Flush** (card value is the highest card points of the five cards)
- **Straight** (card value is the highest card points of the five sequential cards)
- **Three of a kind** (card value is the card points of the same three cards)
- **Pair** (card value is the highest card points of any pair)
- **High Card** (card value is the highest card points of the five different cards)

For example: five cards ("♣ J♣", "♥ 5♥", "♠ J♠", "♦ J♦""♣ J♣") is a Four of a kind (of card value: J); five cards ("♥10♥", "♠ 5♠", "♦10♦""♣10♣", "♥ 5♥") is a Full house (of card value: 10); five cards ("♠ 5♠", "♦ J♦""♣ 5♣", "♥ A♥", "♠ A♠") is a Pair (of card value: A); and so on.
For simplicity purposes, if two five-card poker hands have the same hand-ranking category and the same card value of this category, it is determined to be a tie game, and no further comparison is used for this project.

In your program of simplified Texas holdem game, the following actions will repeated until the user enters "Q" or "q" to quit the program:

1. The 52-standard cards in the deck are created and then shuffled (randomized);
2. Two hole cards are dealt to each of the user player (player 1) and the computer player (player 2) respectively;
3. Five shared cards are dealt face up at the center of the table;
4. Each player combines own two hole cards with five shared cards (seven cards), and generates the best five-card poker hand according to the evaluation of rankings described above and displays on the screen.
5. The best five-card poker hands from the user and the computer player are compared to decide who wins the game, and display the result.
6. Go back to step 1 until the user chooses to quit the program.

This project mainly contains **FOUR** classes: **`Card`**, **`Deck`**, **`Rank`** and **`PokerHand`**. You can download a zip file named `PA1source.zip` from the link (http://www.cs.uky.edu/~yipike/CS216/PA1source.zip) and save the file into the directory you create for Project 1, say **`~/CS216/PA1`**:

```
$ curl -O http://www.cs.uky.edu/~yipike/CS216/PA1source.zip
```

The zip file contains EIGHT files: card.h, card.cpp, deck.h, rank.h, pokerhand.h, testPA1.cpp, SortedLinkedList.h, and SortedLinkedList.cpp.

You will gradually build your project through Lab4, Lab5 and Lab6. The definition of the class named **`Card`** is given to you in the source code. You should finish the definition of classes named **`Deck`** and **`Rank`** in Lab5, and the definition of class named **`PokerHand`** in Lab6 respectively. You can copy the source code for each of the definitions of these classes after you have tested it correctly, to the directory you created for Project 1, say PA1 under CS216 directory:

```
$ cp ~/CS216/Lab5/deck.cpp ~/CS216/PA1/

$ cp ~/CS216/Lab5/rank.cpp ~/CS216/PA1/

$ cp ~/CS216/Lab6/pokerhand.cpp ~/CS216/PA1/
```

Compare to Lab4, the only difference of the class declaration for the Card class in Project 1 is that Card class assigns the friendship to class PokerHand, so that PokerHand class can access the private data members of Card class directly. After you finish Lab5, the rest part of Project 1 is to finish the class definition of PokerHand class, and then you are ready to write your main function.
The PokerHand class is more complicated than other classes you have defined for this project. It has two data members: one is an array of FIVE **`Card`** objects, which represents five-card poker hand; another one is a **`Rank`** object, which represents the hand-ranking (`kind` and `card point`). If you are not familiar with poker game, please refer to the list of hand-ranking on page `1` of this document. There are quite a few public member functions of the PokerHand class, by reading the comments of each function in PokerHand.h, you can provide your own implementation. You may notice that there are three private member functions in PokerHand.h, and what is the reason to define some private member functions? You should make a function **`private`** when you don't need other objects or classes to access the function, when you'll be invoking it from within the class. For PokerHand class, these three private member functions named `sort()`, `isAllOneSuit()` and `isSequence()` are really helper functions, which are useful for the implementations of other public member functions, for example, `sort()` helper function makes every public member function, which is used to decide the hand-ranking, easier to implement by sorting the five cards in decreasing order by card points first, then decide which hand-ranking category it belongs to. However, these helper functions should not be called outside the PokerHand class, that is why they are declared as `private`.

After finish the definition of one class, you can compile the source code with –c option by typing:

```
$ g++ -c card.cpp

$ g++ -c SortedLinkedList.cpp

$ g++ -c deck.cpp

$ g++ -c rank.cpp

$ g++ -c pokerhand.cpp
```

If your definitions for each of FOUR classes have passed the compilation, you are ready to test your source code with `testPA1.cpp`, which you download from `PA1source.zip` file, by typing:

```
$ g++ -c testPA1.cpp

$ g++ -o testPA1 card.o deck.o rank.o SortedLinkedList.o
pokerhand.o testPA1.o
```

The following shows the output when you run your executable program `testPA1` as follows:

```
$ ./testPA1↵

*** Best five-card hand ***
Five cards in order:

♥ A♥

♥ Q♥

♦ Q♦

♠ J♠

♦ 6♦

Its rank is: Pair( Q)

*** Best five-card hand ***
Five cards in order:

♥ Q♥

♥10♥

♥ 8♥

♥ 6♥

♥ 3♥

Its rank is: Flush( Q)

*** Best five-card hand ***
Five cards in order:

♠ Q♠
```

♦ 9♦

♣ 9♣

♥ 9♥

♥ 8♥

Its rank is: ThreeOfAKind(9)

***** Best five-card hand *****
Five cards in order:

♥ J♥

♥10♥

♥ 9♥

♥ 8♥

♥ 7♥

Its rank is: StraightFlush(J)

***** Best five-card hand *****
Five cards in order:

♠ A♠

♣ Q♣

♦ Q♦

♠ Q♠

♥ Q♥

Its rank is: FourOfAKind(Q)

***** Best five-card hand *****
Five cards in order:

♥ Q♥

♥ J♥

♠10♠

♦ 9♦

♥ 8♥

Its rank is: Straight(Q)

***** Best five-card hand *****
Five cards in order:

♥ Q♥

♦ Q♦

♦ 5♦

♣ 5♣

♠ 5♠

Its rank is: FullHouse(5)

The Best five-card hand from all testing cases is:
Five cards in order:

♥ J♥

♥10♥

♥ 9♥

♥ 8♥

♥ 7♥

Its rank is: StraightFlush(J)

Now you are ready to write your own main function to play a *simplified* poker game named Texas holdem, between the user player and the computer player. The following shows a sample output when you run your program as follows:

$ ./CS216PA1↵

The cards in your hand are:

♣ 3♣

♦ 6♦

The FIVE cards on the deck to share are:

****

♠ J♠

♦ 5♦

♣ 8♣

♦ 9♦

♦10♦

****
The cards in computer's hand are:

♦ 8♦

♣ 2♣

Player 1: You
*** Best five-card hand ***
Five cards in order:

♠ J♠

♦10♦

♦ 9♦

♣ 8♣

♦ 5♦

Its rank is: HighCard(J)

Player 2: Computer
*** Best five-card hand ***
Five cards in order:

♦10♦

♦ 9♦

♣ 8♣

♦ 8♦

♦ 5♦

Its rank is: Pair(8)

```
Sorry, the computer wins the game!

Do you want to play poker game again (Press "q" or "Q" to quit
the program) ↵
The cards in your hand are:

♥10♥

♠ 4♠

The FIVE cards on the deck to share are:
**********

♥ 9♥

♦ 4♦

♣ 9♣

♦ 2♦

♠ 9♠

**********
The cards in computer's hand are:

♣ 6♣

♣ 3♣

Player 1: You
*** Best five-card hand ***
Five cards in order:

♠ 9♠

♣ 9♣

♥ 9♥

♦ 4♦

♠ 4♠

Its rank is: FullHouse( 9)

Player 2: Computer
*** Best five-card hand ***
Five cards in order:
```

♠ 9♠

♣ 9♣

♥ 9♥

♦ 4♦

♦ 2♦

Its rank is: ThreeOfAKind(9)

Congratulations, you win the game!

Do you want to play poker game again (Press "q" or "Q" to quit the program) ↵
The cards in your hand are:

♦ 3♦

♦10♦

The FIVE cards on the deck to share are:
****

♥ 2♥

♥10♥

♣ 6♣

♥ 6♥

♥ 7♥

****
The cards in computer's hand are:

♠ K♠

♥ J♥

Player 1: You
*** Best five-card hand ***
Five cards in order:

♥10♥

♦10♦

♥ 7♥

♥ 6♥

♣ 6♣

Its rank is: Pair(10)

Player 2: Computer
*** Best five-card hand ***
Five cards in order:

♥ J♥

♥10♥

♥ 7♥

♥ 6♥

♥ 2♥

Its rank is: Flush(J)

Sorry, the computer wins the game!

Do you want to play poker game again (Press "q" or "Q" to quit the program) ↵
The cards in your hand are:

♥ 6♥

♥ 7♥

The FIVE cards on the deck to share are:
****

♣ 7♣

♣ 8♣

♠ 6♠

♦ 8♦

♦ 5♦

****
The cards in computer's hand are:

```
♥ Q♥

♦ J♦

Player 1: You
*** Best five-card hand ***
Five cards in order:

♦ 8♦

♣ 8♣

♣ 7♣

♠ 6♠

♦ 5♦

Its rank is: Pair( 8)

Player 2: Computer
*** Best five-card hand ***
Five cards in order:

♦ 8♦

♣ 8♣

♣ 7♣

♠ 6♠

♦ 5♦

Its rank is: Pair( 8)

It is a tie game!

Do you want to play poker game again (Press "q" or "Q" to quit
the program) Q↵
Thank you for using this program!
```

Note the blue part is the user input, ↵ represents the "return" key.

Part 2 makefile.

Create a make file (name it makefile) to build your C++ program. In the makefile, name the executable program `CS216PA1` (NO .exe).

- Create a dependency line, which helps you to build the target named `testPA1`. When your TA types "`make testPA1`", your `makefile` should build the executable program containing `testPA1.cpp` and your definition of classes.
- Create the default target, which helps you build the executable program `CS216PA1` and efficiently track the dependency among source files.
- Create a target named `Clean`, which help you to clean up the mess.

After testing your program with both testing cases in **`testPA1.cpp`** and the main function you write in **`PA1.cpp`**, please follow the following instruction to prepare your submission:

For all files:

- Test on your Virtual Machine. If (for whatever reason) it doesn't work there, you lose points. If your program cannot pass compilation, then you get zero credit.
- Your program is unzipped then compiled/linked by executing the command:

  `$ make`

  therefore:
  - All needed files are supplied and assumed to be on the current directory
  - No hardcoded directory names in the `.h` file names
  - Create your makefile (file name makefile)
  - In the makefile, name the created program CS216PA1
- Each class has a separate specification file (classname.h) and implementation file (classname.cpp) unless two classes have very closed relationship
- Comments for every file you write, lay out your source file in a readable style
- Name your zip file `CS216PA1.zip` and zip using the following command:

  `$ zip CS216PA1.zip makefile *.cpp *.h`
  `(or list the individual files)`

  NOT!!!

  `$zip CS216PA1.zip /PA1/*`

  Zip from the directory where the files are, not a parent directory.
  DO not zip either from your home directory or from CS216 directory.
  This creates a subdirectory when your zip file is unzipped, and will cause the grader extra work. When you cause the grader extra work, you lose points.

Submission:

Open the link to Canvas LMS (https://uk.instructure.com/), and log in to your account using your linkblue user id and password. Please submit your file (`CS216PA1.zip`) through the submission link for "**`Project 1`**". It is your responsibility to check whether your submission is successful. If it is not, submit it again till you get the confirmation that your submission is successful.

(Late assignment will be reduced 10% for each day that is late. The assignment will not be graded (you will receive zero) if it is more than 3 days late. Note that a weekend counts just as regular days. For example, if an assignment is due Friday and is turned in Monday, it is 3 days late.)

Always read the grading sheet for each project assignment. It lists typical errors. Check for these errors before submitting your source code. Please note that your C++ program must compile in order to be graded. If your program cannot pass the compilation, you will get 0 point.

(The grading sheet is on the next page.)

Grading Sheet for Project Assignment 1

Total: 100 points.

These are example errors. There are other ways to lose points. C++ programs must compile in order to be graded	Points	Deducted Points
C++ Program	45	
User is allowed to repeatedly play the game till press Q or q to quit the program	3	
The output of your program is user friendly (clear and similar to the sample output)	5	
Correctly decide the ranking of the five-card poker hand (ranking category and card point)	5	
Correctly decide and display which player wins the game or it is a tie game	5	
Provide the correct definition of PokerHand class	25	
Provide separate .cpp files and header files for class(es).	2	
makefile	15	
Generate the executable file correctly	4	
Allow "make testPA1" to build the target testPA1 for testing purpose	4	
The dependency lines correctly track of files' dependencies	2	
The command lines correctly create targets	2	
Allow "make clean" to clean up the mess	3	
Testing	28	
Pass all the testing cases in testPA1.cpp	7*4=28	
Miscellaneous errors, or did not follow the directions in the project assignment, examples:	5	
-2 did not zip file or used tar or gzip instead	2	
-2 created subdirectory when unzipped	2	
-1 wrong names	1	
There may be other errors in this category		
Comments	7	
C++ program comments	6	
Makefile comments	1	
C++ program: -6 non -4 only a few		
Your Score for PA1		