Project 4: CSC 24400
BlackJack

Description:

For your fourth programming assignment, you are to write a program that will implement the game of Blackjack. Your program must use the header files and source code files provided, to which you will have to implement some of the class member functions, as well as the functions to implement the playing rules for the game. It is also to include any other functions you deem necessary. This program will not handle betting, and therefore will not handle the rules related to splitting, doubling, insurance, or surrender.

BlackJack Rules

Blackjack, also known as twenty-one, is a comparing card game between one or more players and a dealer, where each player in turn competes against the dealer, but players do not play against each other. It is played with one or more decks of 52 cards; most commonly 6 or 8 decks in casinos.

The objective of the game is to beat the dealer in one of the following ways:

- Get 21 points on the player's first two cards (called a "blackjack" or "natural"), without a dealer blackjack (if the dealer also gets a blackjack, then the hand is a tie/draw),
- Reach a final score higher than the dealer without busting (exceeding 21), or
- Let the dealer draw additional cards until their hand exceeds 21 (dealer busts)

In American casinos, players are each dealt two cards, face up. In the U.S., the dealer is also dealt two cards, normally one up (exposed) and one down (hidden – called the hole card). The value of cards two through ten is their face value (2 through 10). Face cards (Jack, Queen, and King) are all worth ten. Aces can be worth one or eleven. A hand's value is the sum of the card values. Players are allowed to draw additional cards (called a hit) to improve their hands. A hand with an ace valued as 11 is called "soft", meaning that the hand will not bust by taking an additional card; the value of the ace will become one to prevent the hand from exceeding 21. Otherwise, the hand is "hard". Once the player is satisfied, he stands (takes no more cards), and play moves on to the next player.

Once all the players have completed their hands, it is the dealer’s turn. The dealer hand will not be completed if all players have either busted or received Blackjacks. The dealer then reveals the hidden card and must hit until the cards total 17 or more points. In this program, the dealer will stand on a soft 17 (in other words will not take another card when the total is 17). At many casino tables the dealer also hits on a "soft" 17, i.e. a hand containing an ace and one or more other cards totaling six.) Players win by not busting and having a total higher than the dealer, or not busting and having the dealer bust, or getting a blackjack without the dealer getting a blackjack. If the player and dealer have the same total (not counting blackjacks), this is a tie, called a "push". Otherwise, the dealer wins.

In this program, there will only be one player (the user), and the dealer (the program). Your program should allow game play with anywhere from 1 deck up to 20 decks. A device called the “shoe” holds the decks while cards are being dealt during game play.

INPUT:

There are two sources of input for this program. First, your program must read the command line to determine if there are options for the program. The only supported option is:

-s <number>

where the number must be an integer, and represents the seed value for seeding the random number generator. If the option is not –s, or the option argument is not a long integer (i.e. it's a float or a string, etc.), then your program should print the usage message

Usage: Blackjack [-s <number>]

and then exit/terminate. The brackets indicate the contents inside the brackets are optional. The angle brackets indicate that a legal long number is supposed to be supplied as the argument to the –s option. All other input will come from the user's terminal (i.e., from standard-in – or cin). At the beginning of the program, you should prompt the user for the number of decks of cards to play with. After that, the program should prompt the user if he wants to play a hand of Blackjack. If the user answers "yes", or "y", or any case combination thereof, then the program should deal out the player's and dealer's hands and begin game play. If the user answers "no", or "n" or any case combination thereof, then the program should print to the screen summary statistics about how many hands were played, how many hands were won by the dealer, and how many hands were won by the player. If the user provides any other input, then it should be treated as invalid input, and the player should be repeatedly prompted whether he wants to play a hand until he enters legal input.

The only other input your program will have to handle is to ask the player whether he wants to hit or stand while it is his turn to play. If the user answers "hit", or "h", or any case combination thereof, then the program should deal another card to the player. If the user answers "stand", or "s" or any case combination thereof, then the program should stop dealing cards to the player, and the player's turn ends.

PROCESSING:

1. Print the output header to the terminal screen.
2. Start by checking to see if there is a command line option, and if so process it.
    a. If the option is –s and its argument is a legal long number, then use that number for the seed value
    b. If the option is not –s, display a usage message and terminate
    c. If the option is –s, but the argument is not a legal long number (i.e., it is a float or a character or a string), display a usage message and terminate
3. Determine the seed value to use:
    a. If a legal seed value was provided on the command line, then use that value for the seed
    b. Otherwise, obtain the current system time using the **`gettimeofday()`** library function (found in the header "sys/time.h" You should read the linux man pages, or search online, to see how to use this function.
4. Seed the random number generator by calling the function **`srand()`** with the seed value determined in step 2.

5. Prompt the user for how many decks of cards to use. The user should provide an integer number that is between 1 and 20. Any input other than the integers 1 through 20 should be considered invalid, and the program should repeatedly prompt for the number of decks to use until he enters a legal integer between 1 and 20.
6. Create a deck of cards, and then append that deck to the shoe the number of times specified. The shoe should end up containing the number of decks specified.
7. Shuffle the cards in the shoe using the random number generator **`rand()`** to determine the order of the cards in the shoe.
8. Prompt the user if he wants to play a hand of Blackjack.
    a. If the user answers "yes", or "y", or any case combination thereof, then the program should call a function **`PlayHand()`** to play the hand of the game (see below).
    b. If the user answers "no", or "n" or any case combination thereof, then the program should print to the screen summary statistics about how many hands were played, how many hands were won by the dealer, and how many hands were won by the player, and terminate.
    c. If the user provides any other input, then it should be treated as invalid input, and the player should be repeatedly prompted whether he wants to play a hand until he enters legal input.
9. Just prior to program termination, your program should print the output footer to the screen.

PlayHand() Processing

1. Deal the initial cards to the dealer and the player
    a. Check if conditions are satisfied for shuffling the deck, and if so, then shuffle the deck
        i. If there are less than 52 undealt cards remaining in the shoe, then reshuffle
        ii. If the number of cards left in the shoe is less than 30% of the shoe's original size, then reshuffle
        iii. To reshuffle the deck, set the shoe's size back to its original size, and then call shuffle()
    b. Deal the first card to the player
    c. Deal the second card to the dealer
    d. Show the hand to the screen
    e. Deal the third card to the player
    f. Deal the fourth card to the dealer
    g. Show the hand to the screen (but don't show the dealers hole card – display "XX" for the hole card)
2. Check if either the dealer or the player has a blackjack
    a. If both have a blackjack, then no one wins the hand
    b. Otherwise if the player has a blackjack, he wins the hand
    c. Otherwise if the dealer has a blackjack, he wins the hand
    d. In any case, show the hands a final time (showing the hole card this time)
3. Execute the rules of the PlayerTurn
4. Determine if the player went bust
    a. if so, he loses the game – show the hand one final time (showing the hole card this time) and the **`PlayHand()`** function should end
    b. otherwise the turn switches to the dealer
5. Execute the rules of the DealerTurn
6. Determine if the Dealer went bust
    a. if so, he loses the game – show the hand one final time (showing the hole card this time) and the **`PlayHand()`** function should end

    b. otherwise the a winner needs to be determined
7. Determine the winner of the game
    a. If the Dealers card values add up to a higher number than the players, the dealer wins
    b. If the players card values add up to a higher number than the dealers, the player wins
    c. If the dealer and player have a tie value, then the game is a draw/tie – no one wins
8. Prompt the user if he wants to play *another* hand of Blackjack.
    a. If the user answers "yes", or "y", or any case combination thereof, then the program should call a go back to step 1.
    b. If the user answers "no", or "n" or any case combination thereof, then the program should go to step 9.
    c. If the user provides any other input, then it should be treated as invalid input, and the player should be repeatedly prompted whether he wants to play a hand until he enters legal input.
9. Print to the screen summary statistics about how many hands were played, how many hands were won by the dealer, and how many hands were won by the player, and terminate.

PlayerTurn Processing

1. PlayerTurn - Repeatedly prompt the player whether he wants to hit or stand.
    a. If the user answers "hit", or "h", or any case combination thereof, then the program should
        i. deal a card to the player (place a copy of the card in the player's hand) and remove it from the shoe.
        ii. Show the hands to the screen (but don't show the hole card)
        iii. Determine if the player busted – if so player loses and hand is terminated (be sure to show the hand again including the hole card)
    b. If the user answers "stand", or "s" or any case combination thereof, then the program should end the players turn, and go to step 11.
    c. If the user provides any other input, then it should be treated as invalid input, and the player should be repeatedly prompted whether he wants to play a hand until he enters legal input.

DealerTurn Processing

1. DealerTurn - The dealer will always hit (deal a card to his hand) as long as the sum of the cards is less than 17, and will stand as soon as his card sum is equal to or greater than 17 (the "hard hit" rule).

Other Processing Requirements

1. Computing a hand's sum:
    a. If the hand contains no aces, the sum is the sum of the card values.
    b. If the hand contains aces, then aces count as 11 points, unless the sum exceeds 21.
    c. If the hand contains aces and the sum exceeds 21, then aces are converted as follows
        i. each successive ace (one at a time) is converted to a value of 1 until the sum is less than 21, then stop
        ii. if all aces end up being converted to 1's, then the sum is simply the sum of all the card point values, even if it exceeds 21.
2. Using srand and rand
    a. Include the cstdlib header file (#include <cstdlib>)
    b. Call **`srand()`** once, at the beginning of the program, to initialize the random number generator to the specified seed value (pass the long int value as input to **`srand()`**).

c. Make calls to rand() to generate *n* random numbers between 0 and *n*-1 (where *n* is the number of cards in the deck) with no duplicates (if a random number is seen more than once, all successive occurrences should be discarded, i.e., not used). This sequence of numbers determines the new order of the cards in the deck. Card 0 moves to the index position identified by the first random number, card 1 moves to the index position identified by the second random number, and so on, until all cards have been rearranged.

Note: You should have two functions to show the hands. The first will print all of the cards in each hand. The second will print all of the cards in both hands, except that the dealers hole card will be shown as "XX" so that the player cannot see his hole card.

OUTPUT:

All output from this program must be generated to the screen/terminal. The format of the output should conform to the example below. Not all possible hands can be displayed here, but the output generated should always follow the same pattern. When displaying cards, the cards face and suit should always be printed right justified in a field width of 3 characters, but the left most character is only used when displaying a 10 card. There should always be 1 space between the fields to print the cards in a hand. Also, note that the program should show the visible sum of the cards on the same line as the label which identifies whose hand is being displayed. There should be 5 spaces after the player's label and before the sum label. In addition, there should be 65 '=' characters in a line which separates each hand being displayed, and there should be two such lines when the turn changes from the player to the dealer and when a new hand is started. There should be two rows of 65 '*" each after the player answers no to playing another hand. This should be followed by the summary statistics: the number of hands played, the number of hands the dealer won, and the number of hands the player won.

Note: when debugging, it is often useful to use the same seed value for srand, as this will cause the same shuffling of the deck, which can be useful in repeating the game play so that you can determine if changes to your code have had the desired effect.

Assuming you execute your program as:

> Blackjack –s 1

Then your output should look something like this:

```
Michelle Lynn CSC 24400 Section 11
Fall 2017 Assignment #4
-----------------------------------------------------------------------
Seed value is: 1
How many decks do you wish to play with (max 20)?
20
Do you wish to play a hand of blackjack? y
=================================================================
Dealer Hand:     sum: 10
10H

Player Hand:     sum: 10
 QS
```

```
=================================================================
Dealer Hand:     sum: 10
10H  XX

Player Hand:     sum: 19
 QS  9H

Hit or Stand: s
=================================================================
=================================================================
Dealer Hand:     sum: 20
10H  KC

Player Hand:     sum: 19
 QS  9H

Dealer Stands
Dealer has winning hand!
Dealer Wins!
Do you wish to play a hand of blackjack? y
=================================================================
Dealer Hand:     sum: 9
 9H

Player Hand:     sum: 2
 2D

=================================================================
Dealer Hand:     sum: 9
 9H  XX

Player Hand:     sum: 12
 2D  JD

Hit or Stand: h
Player Hits
=================================================================
Dealer Hand:     sum: 9
 9H  XX

Player Hand:     sum: 15
 2D  JD  3C

Hit or Stand: h
Player Hits
=================================================================
Dealer Hand:     sum: 9
 9H  XX

Player Hand:     sum: 23
 2D  JD  3C  8C

=================================================================
=================================================================
Dealer Hand:     sum: 19
 9H  KH
```

```
Player Hand:     sum: 23
 2D  JD  3C  8C

Player busted (went over 21)!
Player loses
Do you wish to play a hand of blackjack? y
=================================================================
Dealer Hand:     sum: 7
 7D

Player Hand:     sum: 8
 8C

=================================================================
Dealer Hand:     sum: 7
 7D  XX

Player Hand:     sum: 13
 8C  5H

Hit or Stand: h
Player Hits
=================================================================
Dealer Hand:     sum: 7
 7D  XX

Player Hand:     sum: 19
 8C  5H  6H

Hit or Stand: s
=================================================================
=================================================================
Dealer Hand:     sum: 12
 7D  5H

Player Hand:     sum: 19
 8C  5H  6H

Dealer Hits
=================================================================
Dealer Hand:     sum: 16
 7D  5H  4D

Player Hand:     sum: 19
 8C  5H  6H

Dealer Hits
=================================================================
Dealer Hand:     sum: 26
 7D  5H  4D  KC

Player Hand:     sum: 19
 8C  5H  6H

Dealer busted (went over 21)!
Player Wins!
Do you wish to play a hand of blackjack? y
```

```
=================================================================
Dealer Hand:     sum: 10
 KH

Player Hand:     sum: 3
 3S

=================================================================
Dealer Hand:     sum: 10
 KH  XX

Player Hand:     sum: 7
 3S  4H

Hit or Stand: h
Player Hits
=================================================================
Dealer Hand:     sum: 10
 KH  XX

Player Hand:     sum: 13
 3S  4H  6D

Hit or Stand: h
Player Hits
=================================================================
Dealer Hand:     sum: 10
 KH  XX

Player Hand:     sum: 22
 3S  4H  6D  9H

=================================================================
=================================================================
Dealer Hand:     sum: 13
 KH  3S

Player Hand:     sum: 22
 3S  4H  6D  9H

Player busted (went over 21)!
Player loses
Do you wish to play a hand of blackjack? y
=================================================================
Dealer Hand:     sum: 8
 8H

Player Hand:     sum: 11
 AS

=================================================================
Dealer Hand:     sum: 8
 8H  XX

Player Hand:     sum: 21
 AS 10H
```

```
=================================================================
=================================================================
Dealer Hand:     sum: 19
 8H  AC

Player Hand:     sum: 21
 AS 10H

**** Player has BlackJack - Player Wins ****
Do you wish to play a hand of blackjack? y
=================================================================
Dealer Hand:     sum: 9
 9S

Player Hand:     sum: 8
 8C

=================================================================
Dealer Hand:     sum: 9
 9S  XX

Player Hand:     sum: 10
 8C  2S

Hit or Stand: h
Player Hits
=================================================================
Dealer Hand:     sum: 9
 9S  XX

Player Hand:     sum: 20
 8C  2S  JH

Hit or Stand: s
=================================================================
=================================================================
Dealer Hand:     sum: 19
 9S  QH

Player Hand:     sum: 20
 8C  2S  JH

Dealer Stands
Player has winning hand!
Player Wins!
Do you wish to play a hand of blackjack? y
=================================================================
Dealer Hand:     sum: 3
 3C

Player Hand:     sum: 11
 AD

=================================================================
Dealer Hand:     sum: 3
 3C  XX
```

```
Player Hand:     sum: 12
 AD  AS

Hit or Stand: h
Player Hits
=================================================================
Dealer Hand:     sum: 3
 3C  XX

Player Hand:     sum: 15
 AD  AS  3S

Hit or Stand: h
Player Hits
=================================================================
Dealer Hand:     sum: 3
 3C  XX

Player Hand:     sum: 15
 AD  AS  3S 10H

Hit or Stand: h
Player Hits
=================================================================
Dealer Hand:     sum: 3
 3C  XX

Player Hand:     sum: 25
 AD  AS  3S 10H  JS

=================================================================
=================================================================
Dealer Hand:     sum: 10
 3C  7S

Player Hand:     sum: 25
 AD  AS  3S 10H  JS

Player busted (went over 21)!
Player loses
Do you wish to play a hand of blackjack? y
=================================================================
Dealer Hand:     sum: 3
 3C

Player Hand:     sum: 6
 6D

=================================================================
Dealer Hand:     sum: 3
 3C  XX

Player Hand:     sum: 17
 6D  AH

Hit or Stand: s
=================================================================
=================================================================
Dealer Hand:     sum: 7
 3C  4H

Player Hand:     sum: 17
 6D  AH

Dealer Hits
=================================================================
Dealer Hand:     sum: 14
 3C  4H  7H

Player Hand:     sum: 17
 6D  AH

Dealer Hits
=================================================================
Dealer Hand:     sum: 24
 3C  4H  7H  KC

Player Hand:     sum: 17
 6D  AH

Dealer busted (went over 21)!
Player Wins!
Do you wish to play a hand of blackjack? y
=================================================================
Dealer Hand:     sum: 6
 6S

Player Hand:     sum: 11
 AH

=================================================================
Dealer Hand:     sum: 6
 6S  XX

Player Hand:     sum: 20
 AH  9C

Hit or Stand: s
=================================================================
=================================================================
Dealer Hand:     sum: 11
 6S  5H

Player Hand:     sum: 20
 AH  9C

Dealer Hits
=================================================================
Dealer Hand:     sum: 12
 6S  5H  AD

Player Hand:     sum: 20
 AH  9C
```

```
Dealer Hits
=================================================================
Dealer Hand:     sum: 18
 6S  5H  AD  6H

Player Hand:     sum: 20
 AH  9C

Dealer Stands
Player has winning hand!
Player Wins!
Do you wish to play a hand of blackjack? y
=================================================================
Dealer Hand:     sum: 11
 AS

Player Hand:     sum: 4
 4H

=================================================================
Dealer Hand:     sum: 11
 AS  XX

Player Hand:     sum: 9
 4H  5S

=================================================================
=================================================================
Dealer Hand:     sum: 21
 AS 10S

Player Hand:     sum: 9
 4H  5S

**** Dealer has BlackJack - Dealer Wins ****
Do you wish to play a hand of blackjack? y
=================================================================
Dealer Hand:     sum: 9
 9S

Player Hand:     sum: 10
 JD

=================================================================
Dealer Hand:     sum: 9
 9S  XX

Player Hand:     sum: 16
 JD  6H

Hit or Stand: h
Player Hits
=================================================================
Dealer Hand:     sum: 9
 9S  XX

Player Hand:     sum: 17
```

```
 JD  6H  AS

Hit or Stand: s
=================================================================
=================================================================
Dealer Hand:     sum: 19
 9S 10S

Player Hand:     sum: 17
 JD  6H  AS

Dealer Stands
Dealer has winning hand!
Dealer Wins!
Do you wish to play a hand of blackjack? n
*****************************************************************
*****************************************************************
A total of 11 hands were played
Dealer won 6 hands
Player won 5 hands

 ---------------------------------
|      END OF OUTPUT              |
 ---------------------------------
```

Don't forget to print the header and the footer!

Code Provided
The class declarations for Card, Deck, and Hand are being provided to you, as well as the implementation of class Card (file Card.cpp). The class Card is essentially the same as that used in the in-class exercise. The class declaration for class Card is shown:

```
class Card {
   int value;
   string face;
   string suit;

public:
   Card();
   Card(string f, string s);
   void setCardPoker(string f, string s);
   void setCardBlackJack(string f, string s);
   int getValue();
   string getFace();
   string getSuit();
   void setFace(string f);
   void setSuit(string s);
   void Display();
};
```

For class Deck, you are being provided with the code for initBlackJack(), initPoker(), ShowMemory(), Shuffle(), and Display(); the code is the same as that used in the in-class example covering copy constructors, operator=, and destructors. You will have to write the code for the class Deck member functions Append(), DrawCard(), getSize(), ResetSize(), Deck(), Deck(Deck &D), operator=(), and ~Deck().

```
class Deck {
   int size;
   Card *cptr;

public:
   void initBlackJack();
   void initPoker();
   void ShowMemory();
   void Shuffle();
   void Display();
   void Append(Deck &D);
   Card DrawCard();
   int getSize();
   void ResetSize(int newsize);

   Deck();
   Deck(Deck &D);
   Deck & operator=(Deck &D);
   ~Deck();
};
```

The class Hand is similar to the class Deck, in that it can hold cards. However, instead of holding a full deck of cards, it will hold only the cards in a game players hand. As such, it will need to be able to display the cards in the hand, as well a display the cards in the hand when there is a hole card (face down card). Class Hand will also have to be able to add individual cards to the set of cards pointed to by cptr. When a card is drawn from the shoe (a large Deck), you should append it to the Hand of the player who was dealt the card. There are two member functions for computing the sum of the cards in the hand. These functions should store the result in the BJsum variable, as well as return that sum to the caller. These functions should compute the sum of the Hand using the rules for counting the values of the cards, keeping in mind that if the hand contains one or more aces, then you may have to convert some of them from 11 points to 1 point to keep the hand from busting (going over 21 points). The only difference between these two functions is what happens with a hole card. If the hand contains a hole card, then you can assume it was the last card added to the hand. In computeBJsumNoHole(), the hole card's value should not be added to the sum calculated, as that would give away the value of the hole card. You will have to write the code for all of the class Hand member functions.

```
class Hand {
   int BJsum;
   int size;
   Card *cptr;

public:
   void DisplayNoHole();
   void Display();
   void Append(Card C);
   int  ComputeBJsum();
   int  ComputeBJsumNoHole();
   int  getSize();

   Hand();
   Hand(Hand &H);
   Hand & operator=(Hand &H);
   ~Hand();
};
```

Size of functions
The functions in your program should generally be limited to 20-25 lines of code (not including blank lines, lines consisting only of a comment, or lines containing only a brace). No function should exceed 50 lines of code unless you can justify it (such as a case statement that can't be split, etc. – be sure to provide the justification in comment).

Size of Program
This program is a little larger than the ones we have done so far this semester. It will require about 900-1100 lines of code. Most of the code will be in the class member functions (some of which I will provide and others of which you will have to write). However, around 250-400 lines of code will need to be written in multiple functions which are used by the main function to implement the rules of game play. In total, you can expect to write about 500 to 800 lines of code for this project.

Project Submission:
Create a .zip file containing your .cpp and .h files ONLY (do not include executables or object files please – these take large amounts of disk space). Submit your .zip file to the project submission tool on the Canvas course web page for this assignment. **All submissions must be received by 11:59 p.m. the day they are due in order to receive full credit**.