Programming Assignment 5
Due: Friday, November 16, 11:55PM

Overview: Implement the card game Blackjack, using linear data structures to represent the deck of cards and the cards held by players.

Objective: To gain experience manipulating collections of non-primitive objects.

Description: In the game of blackjack, everyone plays against the dealer. Players are dealt two cards face up and the dealer is dealt one card face up and one card face down. The dealer checks her cards and if they equal 21, the player automatically loses. If the dealer does not have 21, the player repeatedly has the option to ***hit*** (take an additional card) or ***stay*** (end their turn). The player's goal is to get as close to a score of 21, without or going over 21. Going over 21 is called a ***bust***. If the player busts, they automatically lose. When scoring a hand:

- the number cards count at their respective values
- the face cards (jack, queen, king) count as 10 points
- an Ace can count as 1 or 11 points, whichever is most advantageous

Once each player has opted to stay, the dealer takes her turn. The dealer turns up her face down card before playing. The dealer must hit if her hand is less than 17, and must stay once her hand is over 16. If the dealer busts then the player wins, otherwise the person with the higher score wins. If the player and the dealer have the same score, called a ***push***, then neither wins.

You implementation should allow for one to six players.

Coding Requirements:
You will need to develop at least four classes to complete this program.

- **class Card**: Each object represents a single playing card, with attributes to hold a value and a suit.
- **class Deck**: A collection of cards that can be shuffled and dealt. A deck should initially contain the standard 52 unique cards. There must be a method to shuffle the cards in the deck. There must be a method to deal one card from the top of the deck and remove it from the deck.
- **class Hand**: A collection of cards held by the players or dealer. There must be a method to add a card to a hand. It must be possible to iterate through the cards in the hand for display. The must be a method for determining the score of a hand.
- **class Blackjack**: The main controller for the game. It must maintain a deck of cards, a dealer's hand and the player's hands. This object is responsible for moving cards from the deck to the hands, controlling the flow of the game (player and dealer turns), and determing the outcome of the game.

Your program must be buildable using the Code::Blocks IDE and the Gnu compiler. You can develop in any environment you like, but test it on the classroom or lab machines before submitting.

User Interface:

- Use the console for input and output.
- The dealer’s actions will be performed by the computer.
- Player actions should be entered from the keyboard.
- Represent cards by printing the value, followed by the suit.
    - Values: A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
    - Suits: C=clubs, D=diamonds, H=hearts, S=spades
    - The dealer’s face-down card can be represented by ??
- The program must show the state of the game at each turn, by displaying the current cards in each hand.

The allowable user input to this program is fairly limited, perhaps four different single character responses. You should check input carefully and ensure that improper input is not accepted by your program.

The console output for a sample game is given below. Your output does not have to match this format, but it does need to present equivalent information.

```
Dealer:   JC ??
Player 1: 10C 6D
Player 2: AD 3D
Player 1’s turn: (h)it or (s)tay? h

Dealer:   JC ??
Player 1: 10C 6D 7S (busted)
Player 2: AD 3D
Player 2’s turn: (h)it or (s)tay? h

Dealer:   JC ??
Player 1: 10C 6D 7S (busted)
Player 2: AD 3D 6C
Player 2’s turn: (h)it or (s)tay? s
```

```
Dealer:   JC 3H
Player 1: 10C 6D 7S (busted)
Player 2: AD 3D 6C
Dealer’s turn: Dealer hits

Dealer:   JC 3H 6H
Player 1: 10C 6D 7S (busted)
Player 2: AD 3D 6C
Dealer’s turn: Dealer stays

Result:
Dealer:   JC 3H 6H  = 19
Player 1: 10C 6D 7S (busted)  LOSE
Player 2: AD 3D 6C  = 20      WIN
```

Programming Practice Requirements: There should be separate .h and .cpp files for each class. Class definitions should be in *.h files and method implementations should be in *.cpp files. Every file should be appropriately commented with your name and a description of the file at the top.

Project Submission: Using the Sakai assignment feature, you should submit a single ZIP file containing all you source code. As indicated above, all code must be buildable using Code::Blocks and the Gnu compiler.