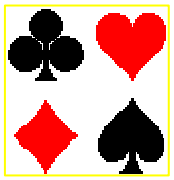

CSCI 2170
OLA 3 (Part A due Tuesday, February 14th
Part B due Tuesday, February 21st)

Write a C++ program which will play a card game called "Hearts". Hearts is a trick taking game in which the objective is to avoid winning tricks containing hearts; the queen of spades is even more to be avoided, yet Jack of Diamond is a welcoming card.

The program plays "Hearts" with four players: computer 1, computer 2, computer 3, and the user. The winner of the game is the person who has the lowest number of points after the game is over. Points in the game are calculated as follows: each card of HEART suit has points: those with face value less than 10 is worth 5 points each, and the rest of the HEART cards worth 10 points each. In addition, the Queen of Spades is worth 100 points, and the club of Jack is worth -100 points.

Each player is dealt 13 cards at the beginning of the game. The cards held by each player should be sorted by suit. The person who holds 2 of CLUB must start the first hand by playing 2 of CLUB. The suit on the first card played on each hand is called the **leading suit** for that hand. Then, the remaining three players must each play a card of the same suit if they have one. If not, they can play a card of any other suit. For each hand, the player with the **highest valued card of the leading suit** must collect the points from that hand. Then, this player starts the next hand in a similar fashion with a new leading suit. Therefore, 13 different "hands" will be played. The game is over after all 13 hands have been played. The player who has the **lowest points** at the end of the game wins.

Your program simulates this game by showing the remaining cards of the **user** (in a table form) when it is his/her turn to play. The user is then asked to select a card to play. At each hand:

- no matter who starts the hand, the four players always play in clockwise rotation.
- display the cards played by each player,
- show points accumulated by each player, and
- **indicate who will lead the next hand**.

The strategy for card selection can be different between the user and the three computers.:

- For the user:
  - if the user is to start a hand, he may select any of the remaining card
  - if the user is to follow a card, e.g., one of the computer player starts the hand, the user should select an appropriate card of the leading suit if he has cards of that suit, or select any card to play if he does not have cards of the leading suit.
  - Your program needs to verify that the user's choice is valid given the rule of the game. If the choice is invalid, prompt the user to enter another choice.
- For each of the three computer players,
  - if it is their turn to start a hand, they will always select to play the first of the remaining cards,
  - if they are to follow a card, they always select the first available card of the leading suit if they have one, or select the first of the remaining cards to play if they do not have cards of the leading suit.

Note:

- create a header file "mytype.h" that contains the definition of the CardStruct only
- include this file in CardClass.h and PlayerClass.h

Part A (50 pts) Due: beginning of class, Tuesday Feb 14th

Add the second class "PlayerClass". The class is used to define the data and related operations /functions of each player. The header file of the class is given below. You are to implement each method according to the description given.

```
#include "mytype.h"
#ifndef  PlayerClass_H
#define PlayerClass_H

class PlayerClass
{
    public:
      // default constructor
      // post-condition: count is assigned 0, score is assigned 0
      PlayerClass();

      // Adds one card to the player's hand.
      // Cards are always added to one's hand by adding it at the end of all cards already in
      // one's hand.
      // pre-condition: player has less than 13 cards
      // post-condition: player has one more card in the hand count is increased by 1
      void       AddCard(CardStruct);

      // prints out the current cards the player has
      // pre-condition:  The player has some cards on hand
      // post-condition: the player's cards are displayed.
      void       DisplayCards() const;

      // select to play the first card that has the suit supplied from the client program
      // if no card can be found that has the suit supplied by the client program, the first card
      // from the hand is played
      // pre-condition: there are >= 1 cards in hand
      // post-condition: a card is returned, count is decremented by 1
      CardStruct  FollowOneCard(suitType s);

      // plays the first card of a round. The top card in the deck is selected
      // pre-condition: there are >= 1 card in the hand
      // post-condition: one card is played/returned, count is decremented by 1
      CardStruct  StartOneHand();

      // Checks to see if the player should lead the
      // first round in the game, e.g., check whether the player has 2 of club
      // pre-condition: the hand is full (have 13 cards)
      // post-condition: if this player has 2 of club, true is returned, otherwise, return false.
      Bool   IsFirstLead() const;

      // The current score of the player is returned
      int        GetScore()  const;

      // the points from the current round is added to the current player's score
```

```
        // pre-condition: a point value is sent in from client
        // post-condition: the player's score is increased by the points from the client program
        void        AddScore(int);

        // Return the number of cards the player has
        // pre-condition: none
        // post-condition:   the number of cards currently in the player's hand is returned
        int         GetCount() const;


        // sorts the cards by suit
        // pre-condition: there are >=1 cards in hand
        // post-condition: the cards are sorted by suit
        void   Sort();

        // plays the card selected by user
        // pre-condition:  the card number selected by the user is supplied
        // post-condition: the card corresponding to the user choice is played/returned the number
        //                      of cards in player's hand is decremented by 1. If this card is in the
        //                     middle of one's hand, all cards following that card are shifted as a result
        //                     of removing the card
        CardStruct  PlaySelectedCard(int choice);


        // Checks to see if the card the user chooses is a valid choice, e.g., whether it matches the
        // leading suit on that hand, if he has any.
        // pre-condition:  the player's choice and the leading suit are supplied
        // post-condition: returns true if
        //        (1) user has cards of leading suit and the choice card is of that suit, or
        //        (2) user does not have card of leading suit
        //     and returns false otherwise
        bool        IsValidChoice(suitType, int)   const;

      private:
        CardStruct   hand[MAX_PLAYER_CARDS];   //  the set of cards held by the player
        int          count;   // keeps record of number of cards player has
        int          score;   // keeps score for the player
    };
    #endif
```

Write the client program to do the following only:

- create a deck of cards in terms of an object of CardClass
- create four players in terms of an array of 4 objects of PlayerClass
- shuffle the cards and deal cards out to the four players in clockwise rotation, one card at a time
- Each player sorts their cards by suit
- The user displays his cards in table form
- The player who has 2 of Club declares the fact

To turn in this portion of the project, do:

```
frank% script log
```

```
frank% pr –n mytype.h
frank% pr –n PlayerClass.h
frank% pr –n PlayerClass.cpp
frank% pr –n ola3.cc
frank% aCC CardClass.cpp PlayerClass.cpp ola3.cc –o run
frank% run
frank% exit
```

Part B: (50 points) Complete the client program by adding code to play the entire "hearts" game. Due beginning of class, Tuesday Feb 21st

To turn in the program :

```
frank% script log
frank% pr –n mytype.h
frank% pr –n CardClass.h
frank% pr –n CardClass.cpp
frank% pr –n PlayerClass.h
frank% pr –n PlayerClass.cpp
frank% pr –n ola3.cc
frank% aCC  CardClass.cpp PlayerClass.cpp ola3.cc –o Run
frank% Run              ← play to completion of a game
frank% Run              ← play to completion of a game
frank% exit
frank% lph log
```