ECE 175: Computer Programming for Engineering Applications
Final Project: *The card game – Crazy Eights*

Due Date: Wednesday April 27, 2016 by 10.59 am, via D2L dropbox

See the last page for late submission policy

Administrative details:

- The project is to be worked in a) a team of two students (maximum) or b) worked individually. *At the demo/check-off time*, your team (***both students MUST be present***) will be asked various questions (similar to your lab assignments, but in more detail). The TA/ULA can ask a question to a specific student to answer, so both members of the team are expected to contribute equally.
  Your team or your name should be on "ECE175_FinalProjectTeam_Individual_SignupSheet_April18_2016.pdf" in Final Project on D2L.
- **Honor code:** Your team is expected to submit your own code, and both members of the team are expected to contribute. You may ask others for advice, and in general discuss the project, but your team should
  WRITE YOUR OWN CODE!
  If any part of the code submitted by different teams is identical, **ALL involved parties will receive zero credit on the entire project and one letter reduction in their final grade**.
  This policy will be very aggressively enforced. ALL submitted code will be checked with a plagiarism detection tool (e.g., http://theory.stanford.edu/~aiken/moss/).

Crazy Eights:

Crazy Eights (https://en.wikipedia.org/wiki/Crazy_Eights) is a card game that can played by two to seven players. For each game, eight cards are dealt to each player (i.e. the dealer deals one card at a time to each player proceeding clockwise (to the left of the dealer) until each player gets 8 cards).

The remaining cards of the deck are placed face down at the center of the table (called *stockpile*). The top card in the stockpile is then turned face up to start the game.

Players discard by matching rank or suit with the top card of the discard pile, starting with the player left of the dealer. If a player is unable to match the rank or suit of the top card of the discard pile and does not have an eight, he or she draws cards from the *stockpile* until getting a playable card.

When a player plays an eight, he or she must declare the suit that the next player is to play; that player must then follow the named suit or play another eight.

As an example: Once the six of clubs (6 ♣) is played (called top card of the discard pile) the next player:

1. can put (on the discard pile) any or all of the other sixes (6 of spades (♠) and/or 6 of ♦ (diamonds) and/or 6 of hearts (♥))
2. can play any of the clubs (♣)
3. can play any 8 (then he/she must declare a suit to play next => The next player must play with the named suit or play another eight)

4. can draw from the stockpile until able to play one of the above. Draw a maximum of 3 cards from the stockpile, if none of the 3 cards is matched, the player then passes.

The game ends as soon as one player has emptied his/her hand OR
there is no card left in the stockpile, the player with minimum number of points on the hand wins.

https://cardgames.io/crazyeights/ - a free online version of a crazy eight game that you can play to get an idea of Crazy Eight

Project requirements:

For this project your team (a group of two students) will write the card game-Crazy Eights, for which you will use a standard deck of 52 cards.

There will be only 2 players, the computer (dealer) and one user. The play will always begin with the user. Play will continue with the computer and the user taking turns until one player empties his/her hand which is a winner.

Card game:

1) To simulate the deck, and each of the players' hands, your team MUST use

a dynamic list of cards with the following type

```
typedef struct card_s {
        char suit;
        int face;
        struct card_s *listp;
} card;
```

each card consists of a suit [clubs (♣),spades (♠),hearts (♥), or diamonds (♦))
a face (1(Ace) – 10, Jacks (J), Queens (Q), and Kings (K))

2) Create a full deck of 52 cards that are in order. In other words, for each of the four suits, the cards should be in order from Ace (1) through King (13).

At the demo time, you will be asked to print the deck that you create.

3) You must then shuffle the deck (52 cards), using the following algorithm:
   (a) For each card in the deck, get a random number in the range of 0 to 50 to be used as the index of the element to swap that card with, i.e.
   if deck[0] holds the Jack of clubs (J ♣) and the random number generated was 24, and deck[24] holds the 9 of diamonds (9♦), then
   after the first swap, deck[0] would hold the 9 of diamonds (9♦) and deck[24] would hold the Jack of clubs (J ♣). You would then proceed to deck[1], find a random index of a card to swap with, and swap those cards, etc.
   (b) Repeat step (a) at least 100 times.

Note: You must seed the random number generator with a call to time() with srand(). [see sec 2.22 Random numbers in your Zyante book]

At the demo time, you will be asked to print the deck after the shuffle is done.

4) After shuffling the deck, you must deal the cards by giving one card to the user, followed by one card to the computer, followed by one card to the user, etc. until **each player get 8 cards.**

   The player's hand is represented as a dynamic list of cards. The list is populated with the cards drawn by the player.

   The dealer's hand is represented as a dynamic list of cards. The list is populated with the cards drawn by the dealer.

   Note:

   a) the card(s) added to each of the player/dealer's hand (drawn from the deck) must be added to that player/dealer's linked list correctly and MUST be removed from the deck.
   b) the cards removed from each of the player/dealer's hand MUST BE deleted from that player/dealer's linked list correctly.

   At the demo time, you will be asked to print both computer's hand and user's hand

5) Your program should then display one card (called, top card of the discard pile)

6) The game will start with a player/user **either** (only one option below can be used)
   a) puts (on the discard pile) any or all of card(s) with the **same FACE** as the face of top card **OR**
   b) puts (on the discard pile) any of card with the **same SUIT** as the SUIT of top card **OR**
   c) puts (on the discard pile) any 8 and then he/she must declare a SUIT to play next **OR**
   d) draws from the stockpile until he/she is able to play one of the above (a – c). Draw a maximum of 3 cards from the stockpile, if none of the 3 cards is matched, the player then passes.

7) Then it is a computer's turn. The computer must put the card(s) on the discard pile as discussed in step 6a – 6d above.

8) The game continues (repeat steps 6 – 8) until one player empties his/her hand OR there is no card left in the stockpile -> the winner is the hand with minimum number of points (add up all the face values of the cards left in each hand).

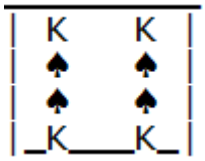
9) At the end of the game, your code should
   - announce the winner
   - **ask whether a user wants to play a new game** (Q/q to quit).
   If yes, your program should repeat steps 2 – 9. (Note 2 -3 may be optional depending on how you write your program.

Grading Rubric – See Grading Rubric pdf file in the Final project module on D2L

Optional Features for extra credits

1) (4 points (⇔ 1 % bonus in the final grade)) **Game statistics:** Keep statistics on the number of games played, the player's winning percentage, the dealer's winning percentage.
   - Create a text file that keeps a user's name, number of wins and number of losses.
   [each line consists of user's name number of wins number of loses]
   You should start with your name and your team member name with some number of wins and losses in the file.
   - At the beginning, ask a user whether he/she wants to create a new user or continue with an existing one.
   - If it is an existing user, the record (previous numbers of wins or loses should be read from the text file)
   - If it is a new user, the record (win or lose) should be updated into the text file at the end of each game (or once the same player decides to quite the game)
   - Once a game is finished, Announce whether a user wins or loses
   Show his/her number of wins and number of loses. If the user is a current one, the current win or loss should be added into his/her record before display and then be kept in the text file (write back into the text file).

2) (2 points max) **Graphics**: Add graphics to your game.
   (+ 1 pt (⇔ 0.25 % bonus in the final grade)) If you can print using ♣, ♠, ♥, ♦
   (+1 pt if you can make the card for each card) such as



   Note: above is an example, you/your team are welcome to create ***better***/different version than this.

3) (2 points (⇔ 0.5 % bonus in the final grade)) **Lazy Shuffling**: Devise a strategy to that allows the dealer to be either very good or very bad card shuffler. You should ask the user on a scale from 1 to 10, how lazy of a shuffler the dealer should be, where 1 is lazy and 10 is thorough.

4) (4 points (⇔ 1 % bonus in the final grade)) Using several functions (may require your program to use a pointer to a pointer) such as (at least the two functions below must be in your program)
   - ***CreateDeck( ) function*** – to create a standard deck of 52 cards, that could be used for any card playing game.
   - ***RemoveCard( ) function*** – to remove a card from a deck or a hand

Make sure that your program

- uses modular programming
- writes well-documented/commented code
- tests your code (come up with suitable test cases)

Note: You are encouraged to make your output look nicer than what is given in the final project module (see Sample code execution). However, your code should print out the same information. You can think of this sample code execution as the minimum requirement of the project.

Late Submission Policy (projects are due prior to 11 AM on Wednesday April 27, 2016):

by 11 AM on Thursday April 28, 2016, deduct 10%
by 11 AM on Friday April 29, 2016, deduct 20%
by 11 AM on Saturday April 30, 2016, deduct 30%
by 11 AM on Sunday May 1, 2016, deduct 40%
by 11 AM on Monday May 2, 2016, deduct 50%
by 11 AM on Tuesday May 3, 2016, deduct 60%

The last date of final project demo is by 5 PM on Wednesday May 4, 2016. After this time the project will NOT be accepted/graded.

If you *submit your code late*, it is ***your team's responsibility to demo your project*** to the TAs/ULAs. (Sign-up sheet for demo will be posted later)