Cairo University
Faculty of Engineering
CMP102/CMPN102

Fall 2019

Data Structures and Algorithms Assignment #1

Crazy-KING Cards Game

Due date
Tue 3/12/2019, 11:55 pm

Regulations

- This is a team-based assignment. Each team consists of 2 students and should submit one solution.
- Assignment should be submitted as one zipped file that contains teaminfo.txt file and code files but without unnecessary file/folders such as debug folders, .sdf, ipch files, .vs folders ,…. etc
- **Cheating penalty:**
  Your code will be checked against other teams code and online code using plagiarism checking tools.
  The penalty of cheating any part of the assignment from any other source is not ONLY taking ZERO in the assignment grade but also taking MINUS (-1.5) from other class work grades, so it is better to deliver an incomplete assignment rather than cheating it.

Requirements:

It is required to make a simplified and modified version of crazy-eight cards game.

Game Description:

Game can be played by 2 to 5 players. The draw pile (***DRW-pile***) starts with the 52 cards faced down.
To start a game **round**, the DRW-pile is ***shuffled*** then 7 cards are dealt from the DRW-pile to each player. Then the top card of the DRW-pile is turned face up and is moved away and placed as the first (bottom) card in the discard pile (***DSC-pile***). Playing a card is to place it face up on top of the DSC-pile

Game objective:

To be the first one to discard all cards to the DSC-pile and to get the winning score.

Game Rules:

☑ Game is played in rounds. Each round continues until a player wins or until DRW-pile runs out of cards 5 times.
☑ At each round, the first player to play is the winner of the last round. If this is the first round ever, a random player is picked to start the game.
☑ ***Round Winner***: is the player who discards all his cards first
☑ Winner score in a round is the total sum of all cards points from other players
☑ **Cards Points**: KING = 30, ACE = 10, QUEEN = 10, and JACK = 10 points. Other cards have points equal to the rank on the card face.
☑ ***Game Winner***: is the player who achieves total score of 500 first.
☑ If DRW-pile runs out of cards at any time:
  - If this is the 5th time DRW-pile runs out of cards, the current round ends and the player with minimum cards total points is the winner of this round.

  - Otherwise, all cards from the DSC-pile except the top card are ***<u>shuffled</u>*** and placed in DRW-pile face down. Only the top card remains in the DSC-pile face up and round continues.

☑ Player should play a matching card (i.e. a **card with same suit or same rank**) for that on top of DSC-pile

☑ If player doesn't have a matching card, he should:

  - Draw a card from DRW-pile and play if matching.
  - Otherwise, draw another card from DRW-pile and play if matching.
  - Otherwise, player loses his turn. (PASS)

☑ KING card breaks the above two rules. (see Action Cards below)

☑ **Action cards:** are cards that cause special actions to be taken in the game.

Action Card	Effect
KING card	It can be played on any card ***regardless of card matching rule***. The player chooses the next suit to be played. You should choose the suit that occurs most at this player's hand.
JACK card	Next player doesn't play (unless he has a KING)
QUEEN card	Next player doesn't play (unless he has a KING) and draws two cards.
ACE card	Play direction is reversed

Program Interface:

Initially, the user enters the number of players (2 to 5) and players' names.

The program should display the game status at each turn as follows:
For the DRW-pile:
The number of remaining cards and how many times it has run out of cards are shown.
For the DSC-pile,
The number of current cards and the top card are shown
For each player, the program should print:

- Player's name and Current Score
- Total sum of cards points the player has at the moment
- All cards the player has at the moment
- Action to be taken by the player (PLAY, DRAW, PASS (loses his turn))

Sample Program output

```
DRW-pile ==>      [30 Cards] [3]

DSC-pile ==>      [7  Cards]      [9|♣]
============================================================
Ahmed       [Score=274]    [2 Cards ]  [Cards Sum = 12]

 [4|♥]   [8|♥]
============================================================
Hesham       [Score=165]    [9 Cards ]  [Cards Sum = 64]

 [A|♥]  [ Q|♦]   [Q|♠]   [4|♠]   [6|♣]   [5|♥]   [9|♥]   [3|♦]   [7|♣]
============================================================
Hazem        [Score=273]    [2 Cards ]  [Cards Sum = 20]
```

```
 [J|♠]  [10|♣]
==================================================================
Omar         [Score=95]     [2 Cards ] [Cards Sum = 16]

 [J|♥]  [6|♠]
==================================================================
TURN of: Hazem ...
ACTION: Hazem will PLAY  [10|♣]
Press any key to advance
```

Then

```
DRW-pile ==>      [30 Cards] [3]

DSC-pile ==>      [8  Cards]    [10|♣]
==================================================================
Ahmed        [Score=274]    [2 Cards ]  [Cards Sum = 12]

 [4|♥]  [8|♥]
==================================================================
Hesham       [Score=165]    [9 Cards ]  [Cards Sum = 64]

 [A|♥]  [Q|♦]  [Q|♠]  [4|♠]  [6|♣]  [5|♥]  [9|♥]  [3|♦]  [7|♣]
==================================================================
Hazem        [Score=273]    [1 Card ]  [Cards Sum = 10]

 [J|♠]
==================================================================
Omar         [Score=95]     [2 Cards ]  [Cards Sum = 16]

 [J|♥]  [6|♠]
==================================================================
TURN of: Omar ...
ACTION: Omar will DRAW a card from DRW-pile
Press any key to advance
```

Classes and Implementation:

You should have a separate class for ***Card***, ***Player***, and ***CardStack*** (a stack of Cards)

Class ***Game*** should be responsible for game playing, scoring, and program interface.

CardStack should be implemented using linked lists. It has five member functions ONLY (push, pop, isEmpty, peek, and toArray). You are not allowed to add any extra member functions.

You should write a global function **Shuffle( )** to be able to shuffle cards of CardStack

Hint: use toArray function to help within Shuffle function.

Both DRW-pile and DSC-pile should be objects of class ***CardStack***. DRW-pile starts with 52 cards and DSC-pile starts with zero cards.

PlayerCards is a class that represents the list of cards that are hold by each player. This should be implemented as a general linked list. Each Player starts with a list of zero cards then seven cards a dealt to each player before playing starts.

Cards should be **moved** between different lists of cards (***CardStack*** and ***PlayerCards***) without creating a new object. ***(MOVE, DON"T COPY)***