Data Structures (2028C) -- Spring 2018 – Homework 3

Topics covered: Working with Stacks and Queues

Homework due: ***Friday, Mar 2 at 1:20PM***

Objective:

The objective of this homework is to create an implementation of a Stack and Queue class and implement those classes in a simple game. The Stack and Queue should both be built with linked lists rather than arrays.

Scenario:

You are creating a new version of the card game, War. In this game, the player will play against the computer. A card will be drawn by each game participant (player and computer). The winner of the round will get to keep both cards. The new rules for this are each player can peek at their card and decide to pull an additional card out of a pile on the side or push this card onto a pile on the side. Should the participant push the card, they will draw a new card from their deck and only be able to play that one card this round. Should the participant pull a card from their side pile, they must play both cards this round. The winner of the round is determined by the sum of the face value of the cards being higher than the opponent. Ties go to the computer. If a participant runs out of cards in their deck, they must play cards from their side pile. If a participant runs out of cards in there deck and side pile, they lose. This game will run from a command line/console window. This needs to be written using C++.

Requirements:

1. Create a deck class that models a queue. This will provide a card from the top and return all cards won to the bottom. This needs to be able to tell the player how many cards they have left. The player can ask how many cards the computer has left in their deck.
2. Create a side pile class that models a stack. This side pile will have a maximum of 5 possible cards in the stack. A player cannot peek at what is in their side pile. A player can add a card to the top or remove a card from the top. This needs a way to tell the player how many cards they have left but not the other player.
3. Create a main method and any necessary support functions to enable a person to play the game. This can be until one participant is out of cards or keep track of number of rounds with the winner of the most rounds winning the game. The style of play (# of rounds or until out of cards) should be declared at the start of the game. At the end of the game, report the winner
4. Create a test case (1 per group member) and execute the test case. Include the completed test case documents with the submission. The test case should **not** be executed by the same person that created it. An example test case can be found here. You are welcome to use any format test case you desire so long as it contains the requisite information. This may not be sufficient testing of your application.

Submission:

Submit all source code files and any required data files in a zip file. Include a write up as a PDF including:

- The name of all group members (minimum 2 members, maximum 4 members).
- Instructions for compiling and running the program including any files or folders that must exist.
- Complete and executed test cases.
- What each group member contributed. If the contributions are not equitable, what portion of the grade each group member should receive.

Submission should be submitted via BlackBoard.

Grading:

1. 20% - Deck class works correctly.
2. 20% - Side Pile class works correctly.
3. 40% - Main functionality works correctly.
4. 10% - Test cases are complete, appropriate, and executed.
5. 10% - Code is well formatted, well commented and follows a reasonable style.

If program fails to compile, the grade will be limited to a max grade of 50%.