Assignment 1

Linked list + Iterators + Stacks

In this assignment you will implement a common card game called solitaire using doubly linked list and iterators.

Solitaire – The Game

Solitaire is a single player game that uses a deck of 52 cards. There are total 4 suits: ♥ Heart, ♠ Spade, ♣ Club, ♦ Diamond (two red and two black) and each containing 13 rank cards (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K).

Game Setup: Shuffle the deck of cards and arrange them in 7 tableau columns. The first tableau column on the left has one card, the second has two cards, the third has three cards, continuing to seven tableau columns. The top card in each tableau column is turned face up, the rest of the cards are face down. The remaining cards go face down in a single pile called the stockpile. There are four foundation piles, one for each suit.

Game Rules: Cards that are face up can be moved from the stockpile or the tableau columns to the foundation piles or to other columns.

To move a card(s) to a tableau column, it must be one less in rank and the opposite colour. For example, if it was a 9 of hearts (red), you could put an 8 of spades or clubs onto it. Multiple cards may be moved from one tableau column to another as long as they maintain the same order (highest to lowest, alternating colours). If all the faced-up cards from a column are moved the topmost faced down card is faced up.

If you get an empty column, you can start a new column with a King. Any new column must be started with a King (or a stack of cards that starts with a King). To get new cards from the stockpile, you turn one cards at a time face up into another pile called the wastepile. You can only play the top card off the wastepile. If you run out of stockpile cards, move all the cards in waste pile to the stockpile and start again.

Each of the four foundation piles (initially empty) can only contain one suit. The foundation piles must be stacked from lowest (Ace) to highest (King). A card can be moved to foundation if it is in the ascending order. Cards in the foundation can be moved back down to the tableau column as long as they obey the column rules (alternate colour and descending order).

The **game is won** when all the cards have been moved to the *4 foundation piles*, with only a single suit being in each of the four foundation piles, and all cards being played in ascending sequence from Ace to King

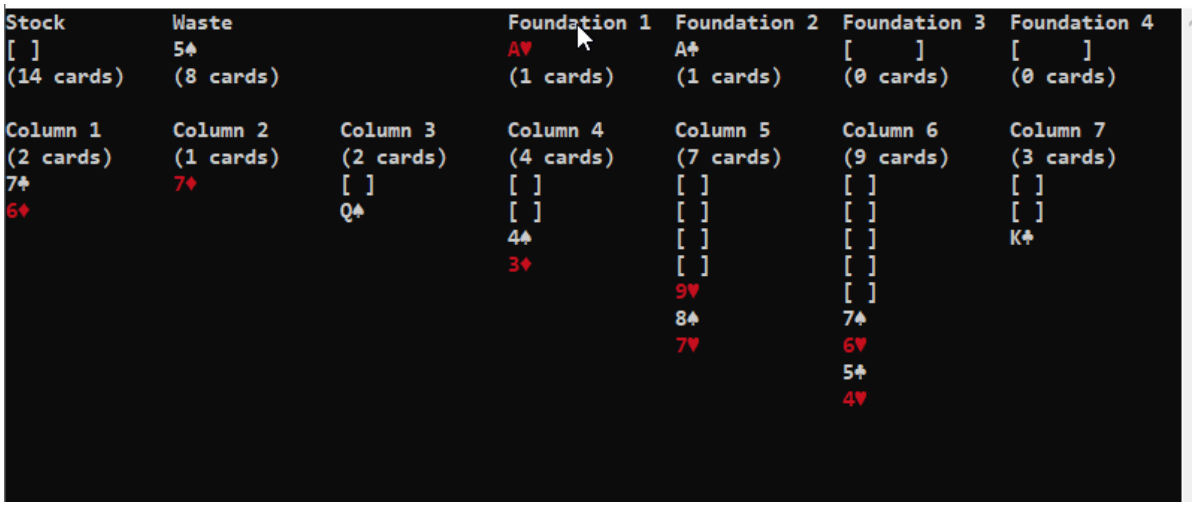
The **game is over** if there are no more moves can be made.

Game Interface:

You will implement a console-based implementation of this game. Following commands will be used for the game play.

- To draw a card from stockpile, use command **s**
- To move a card, use command **m** followed by the *source, destination and number* (for number of cards).
    - **For example: m c6, c1, 2** means move two cards from c6 (column6) to c1 and
    - command **m w, c1, 1** means move 1 card from wastepile to c1.
- To undo a move or draw operation use command **z**

Below is a sample game interface. You can use your own creativity as well.



Implementation:

You will create the following classes to implement this task

List: A doubly linked list and its iterator will be used to implement tableau *columns of cards.*

Stack: A stack using linked list will used to implement foundations, draw and waste piles. It will also be used to implement undo operation.

Card: Each card has a suit and a rank

Command: Each command will store the command and its operands (if any)

Game: The game class will have an array of linked list of cards for tableau columns, stacks of cards for foundations, stock and waste piles. A stack of commands for undo operation. The game will implement all the following functionalities.

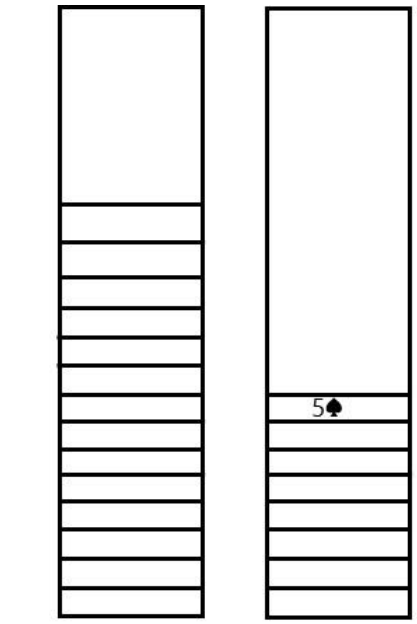
Setup a new game: Shuffle a deck of cards and initialize all the piles, columns and foundations.

Process game move: Get input from the user, parse the input and process the command according to game rules. If its a valid command then update the state of the game accordingly. Otherwise, output an error message.

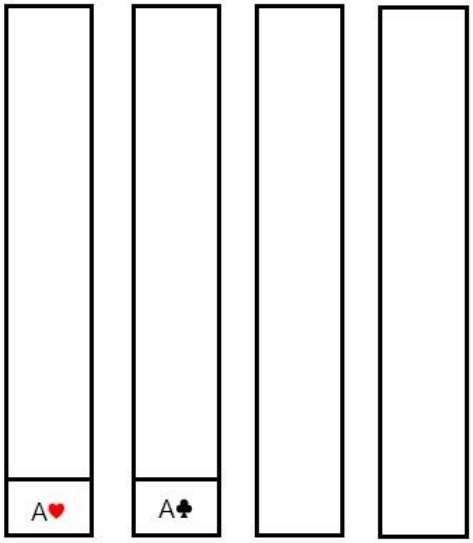
Game Decision: After each move check whether the game has ended or not. If yes, then output the appropriate message and end the game.

Important Note: Make sure that your implementation follows the OOP principals and give the efficient most implementation (the move operation means moving existing objects and not creating new objects and deleting old ones). You may also need to implement functions like moving a sublist from one linked list to the other.

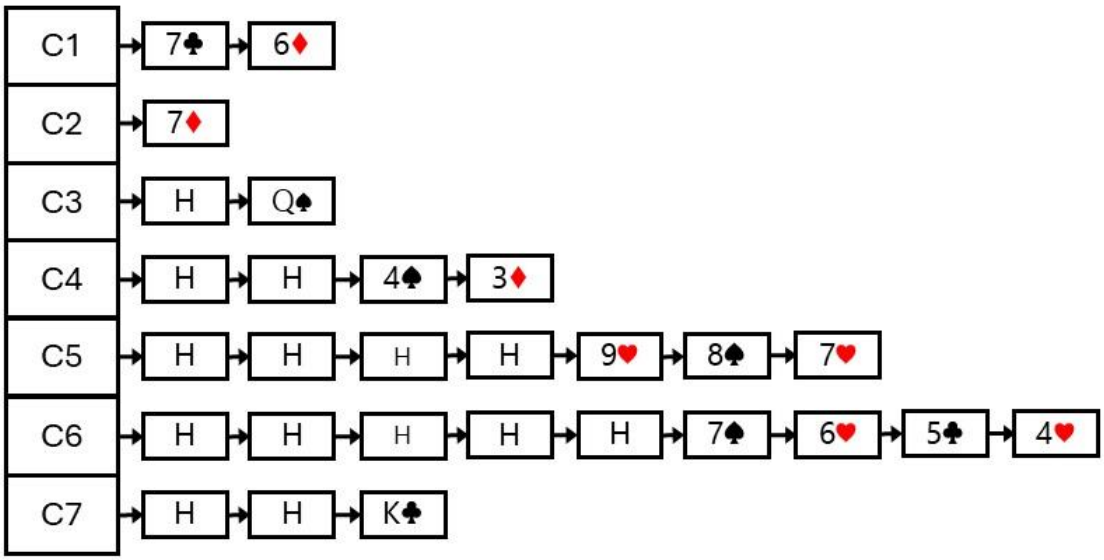
Below is a pictorial representation of the data structures at a particular time.



stockpile wastepile

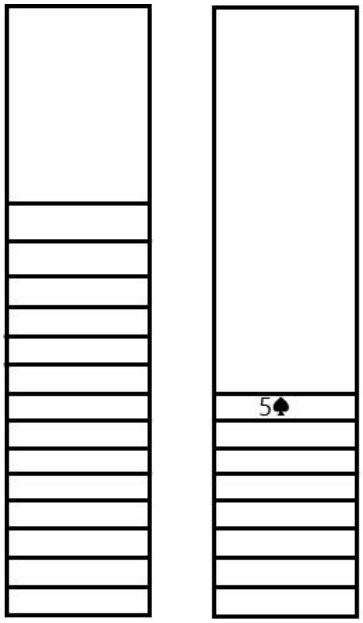


Foundation Stacks

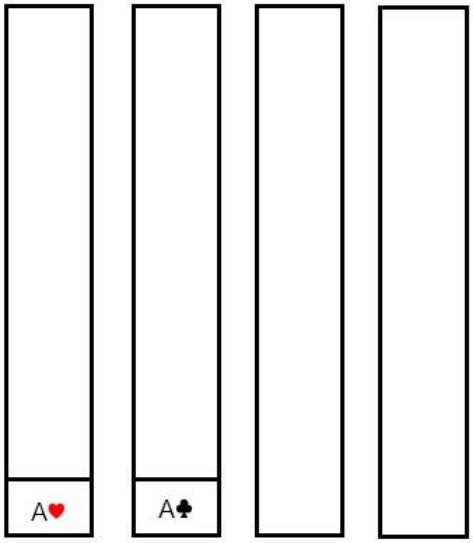


Column Lists

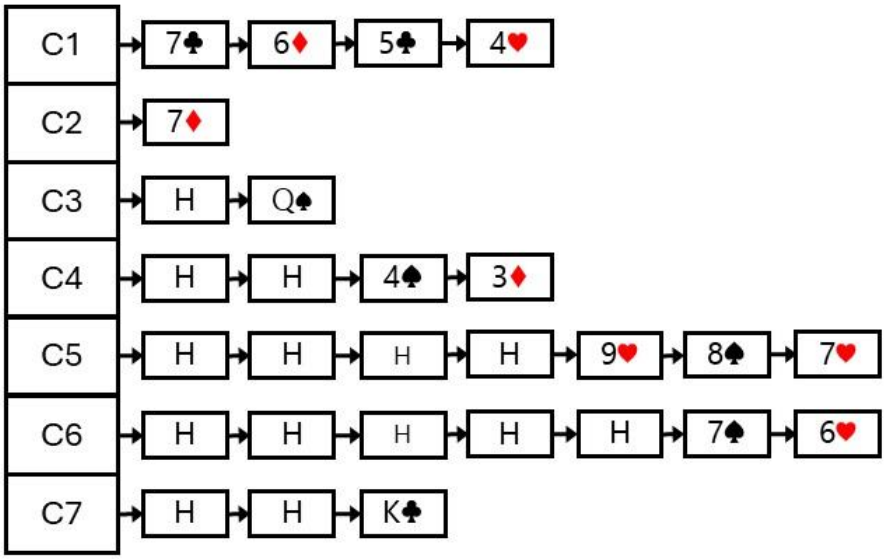
After the command **m c6, c1, 2** the new state of the game is shown in figure below



stockpile wastepile



Foundation Stacks



Column Lists