Assignment 6: Simple Card Game

Dates:

Assigned: 2013-07-18 Thursday
Due: **2013-08-01** Thursday at 11:59 pm

Questions – email rjmfff@mail.umkc.edu

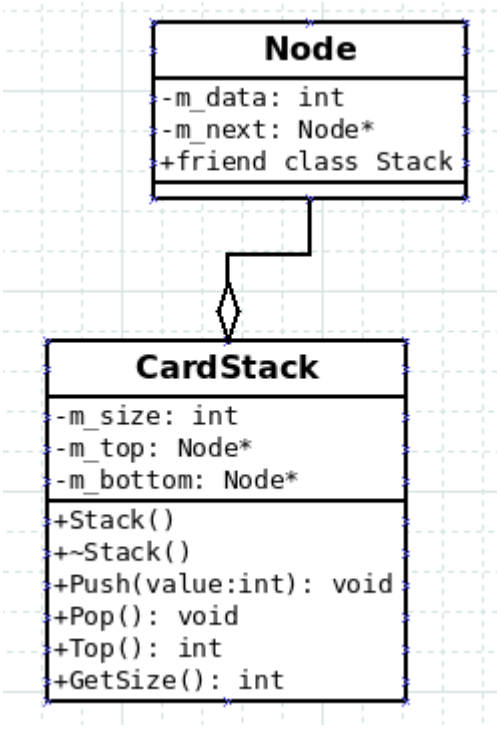
Make sure that your source code is *committed* and *synced* to your student GitHub account. You do not need to turn in the code via Blackboard or anything else.

I will no longer answer questions on assignments on the day they are due, so start it BEFORE Thursday!

Specification:

With this assignment, we are going to implement a Stack with a Linked List. We will generate a stack of 100 cards when the program begins, then continually (automatically) have players pick cards and add up a point value. At the end of the game, whoever has the most points wins.

Class Diagrams



CardStack is both a Stack structure and a Linked List. Remember that with Linked Lists, we have a series of Nodes who point to each other, but we only keep track of the first (and optionally the last) Nodes.

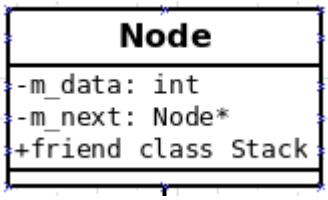
Neither of these classes are templates – we are hard-coding them to store integer data. (m_data)

Step-By-Step

Write the Node class first, and then the CardStack class. If you need a sample Linked List to go off of, look at sample code on the class site...
https://github.com/Moosader/Problem-Solving-and-Programming-II/tree/master/Lecture/17%20Stacks/Sample%20Code/Stack
(Lecture > 17 Stacks > Sample Code > Stack)

Create a **CardStack.h** file in your project.

Node Class

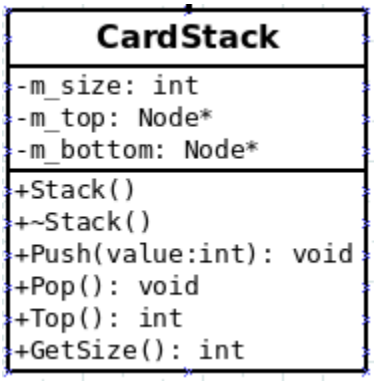


Within CardStack.h, Declare a Node class. The Node class will have two private members:

- Node* m_next
- int m_data

And add **CardStack** as a friend class.

CardStack Class



Within the CardStack class, we will have three private members. Two points to the top and bottom Nodes.

Stack()

Within the Constructor, initialize the top and bottom pointers to NULL and set the size equal to 0.

~Stack()

Make a while loop that keeps Popping items off the stack until the bottom pointer is NULL.

void Push(int value)

With Push, you need to add items to the stack two different ways based on two scenarios:

- Empty List
- Not-Empty List

First, create a new node by dynamically allocating the space. Set up its data, and set the next item equal to NULL.

If the list is empty, then set the top and bottom pointer equal to our new node.

Otherwise, point the top Node's next item to our new node, and then set our top node equal to the new node.

Don't forget to increment the size.

void Pop()

Here, we'll again behave in two different ways based on whether there is only one item in the list, or many items.

If there is only one item in the list, delete that item and set the top and bottom pointer equal to NULL.

Otherwise, if there are multiple items in the list, we need to get the 2nd to top item before we delete the top.

Create a Node pointer that we will use to traverse the stack to the 2nd item:

```
// Find second to last item
		Node* curNode = m_bottom;

		while ( curNode->m_next != m_top )
		{
			curNode = curNode->m_next;
		}
```

After the loop, curNode will be pointing to the second to last item. Set curNode's next item pointer to NULL.

Delete the item that the top pointer is pointing at, and then set the top pointer equal to curNode. (We delete the current top item, and then set the top pointer to a new item).

Don't forget to decrement the size.

int Top()

Return the m_data value of the top item, which is being pointed to by our m_top pointer.

int GetSize()

Return the size of the stack.

In main()

Within main, create an ofstream that opens “Output.txt”, rather than cout-ing our data, we will write it to a file.

Create a CardStack named **cards**, and an array of integers called **players**.
The player array should be of size 5, and initialized to 0. This is how many points each player has.

Before our main loop, we are going to create 100 cards and put them on the deck.

Create a for-loop that iterates 100 times. Each time in the loop, push an integer onto the stack. Have it be a random number.

```
cards.Push( rand() % 8 );
```

You can choose any range you want. The one above goes from 0 to 7.

After we're done setting up the cards, create a while loop that will continue to loop until the cardStack size is 0.

Each time in the while loop, we are going to loop through all 5 of the players.
For each player, we will pull the top card off the stack and give the current player those points. Add the card's points onto the player's current points.

Afterwards, display the current points.

After the while loop, the deck will be empty and all the points will be allocated.
When the game is over, display all players' scores and determine who wins. The person with the most points wins the game.

Make sure everything is output to Output.txt file, and close the file at the end.