SCS214: Data Structures
Assignment-3

Instructions

1- Students will form teams of **2** students (**from the same lab group).**
2- Deadline of submission is **Sunday May 8th at 11:55 pm.**
3- Submission will be on Blackboard.
4- No late submission is allowed.
5- No submission through e-mails.
6- **Please follow the Submission Notes found below**
7- **In case of Cheating you will get a negative grade whether you give the code to someone, take the code from someone/internet, or even send it to someone for any reason.**
8- You have to write clean code and follow a good coding style including choosing meaningful variable names.

Task

1- Implement a class template Node that has the following declaration:

```
template<class T>
class Node
{
    T info;
    Node* next;
public:
    Node(T,  Node* n=0);
    Node* getNext();
    void setNext(Node* );
    T  getInfo ();
    void setInfo (T);
};
```

2- Create a class template SLL, which is a single linked list with the following declaration:

```
template<class T>
class SLL
{
    Node<T> *head, *tail;
public:
    SLL(){head = tail = 0;}
    void addtoHead(T);
    void addtoTail(T);
    T removeFromHead();
    T removeFromTail();
    T getValueAtHead();//a function that returns the value at head without deleting it
    bool isEmpty();
    void clear();
    friend ostream& operator<<(ostream&, const SLL<T>&);
};
```

SCS214: Data Structures
Assignment-3

3- Implement the class Template StackSLL which uses the linked list you created in part (2). The class should have the functions: push , pop , top, isEmpty, and clear. Each of the functions should use the available member functions of SLL.

4- Implement the class Template QueueSLL which uses the linked list you created in part (2). The class should have the functions: enqueue , dequeue , front, isEmpty, and clear. Each of the functions should use the available member functions of SLL.

5- Implement a function **bool chkBalanced(string)** that uses a stack to check if the brackets entered within an expression are balanced. The brackets handled should include { } ,< > , [ ], ( ).
Example expressions: **{ ((x + y) * 63 – a [i+2]) { } }.** Output: **Balanced**
Example expressions: **{ (x + y) * 63 – a [i+2]) { } }.** Output: **Imbalanced**

6- Implement a function **string convertToBinary(int)** that uses a stack to convert a decimal number to a binary number returned as a bit string.
Example input: **218**. Output: **11011010**

7- Implement a function **string convertInfixToPostfix(string)** that uses a stack to convert an infix mathematical expression to a postfix notation. Assume that only 1-digit integers are used within the expression.
Example input: **6 * 3 + 8 / 2** . Output: **6 3 * 8 2 / +**
The algorithm to do such conversion is in Weiss book, Chapter 3: Infix to Postfix Conversion, page 108.

8- Implement a function **int evaluatePostfix(string)** that uses a stack to evaluate a postfix expression and return the value as an integer. Assume that only 1-digit integers are used within the expression.
Example input: **6 3 * 8 2 / +** . Output: **22**

9- Implement a function **bool moveNthElem(Queue<T>SLL&,int n)** that moves a queue element at the specified **N** position to the front of the queue. For example, assuming a queue with the elements {4,6,8,10} and n=3, the 3rd element which has the value 8 will move to the front of the queue. The updated queue will be {8,4,6,10}. The function may use an additional queue.

SCS214: Data Structures
Assignment-3

The function returns true if **N** is within the queue range and false otherwise.

10- Implement a function **void reverseQueue(QueueSLL<T>&)** that reverses the elements of a parameter queue. The function uses a stack for the reverse.

Write a main function that displays a menu as follows:
1- Check for balanced brackets.
2- Convert to binary
3- Convert infix to postfix expression and evaluate.
4- Move Nth element to front in a queue of strings.
5- Reverse elements of a queue of doubles.

For option 1, the string that will be entered will have bracket and non-bracket characters.
For option 2, the decimal number will be entered.
For option 3, an infix notation mathematical expression will be entered. The output should show both postfix notation and the result value of the expression.
For option 4, the queue elements will be entered first, then the position N will be entered and the queue after the update will be shown.
For option 5, the queue elements will be entered first, then the queue after being reversed will be shown.

Submission Notes:

-Submission will be in the form of a zip file.
-Separate your implementation such that:

a. Class Node declaration and implementation will be in file Node.h
b. Class SLL declaration and implementation will be in file SLL.h
c. Class StackSLL declaration and implementation will be in file StackSLL.h
d. Class QueueSLL declaration and implementation will be in file QueueSLL.h
e. Your main function will be in main.cpp

1- Create a folder with 6 files; Node.h, SLL.h, StackSLL.h, QueueSLL.h, main.cpp, project.cbp/project.sln (do not include any other files)
2- Name the folder by your lab group, id1 and id2. Ex.: S5_22222222_33333333
3- Zip your folder (S5_22222222_33333333.zip)

SCS214: Data Structures
Assignment-3

Grading Info:

Node Class	**10**
SLL Class	**25**
StackSLL Class	**10**
QueueSLL Class	**10**
chkBalanced	**20**
convertToBinary	**10**
convertInfixToPostfix	**25**
evaluatePostfix	**20**
moveNthElem	**10**
reverseQueue	**10**
Total	**150**