CS590 homework 2 – Lists, Stacks, and Queues

The due date for this assignment is Friday, Oct 12th, at 11.59pm. This assignment is worth 10% of your final grade.

Any sign of collaboration will result in a 0 and being reported to the Graduate Academic Integrity Board. Late submission policy described in the syllabus will be applied.

Questions (100 points)

1. Design, and implement two algorithms to remove duplicates from an unsorted single Linked-List. You are given the implementation of the single Linked-List in the files *LinkedList.h,* and *LinkedList.cpp****.*** You are also given the skeleton code in *problem1.cpp* where you will implement the requested solution. A test case has been provided in the *main()* function, but feel free to write additional test cases to make sure your code works. (Additional test cases are not requested, and will not be graded).

    a) for the first algorithm, you can use an additional data structure (one of the STL containers – you can add one include to the header file for that container of the STL you decide to use)

    b) for the second algorithm, no additional data structure can be used

2. Design, and implement a C++ class that implements the queue data structure, using two stacks. For the stacks, you are allowed to use the stacks implementation of C++ STL, already included in the source code shared with you. In the *problem2.cpp* file you will find a skeleton of the code where you have to implement the requested solution. A test case has been provided in the *main()* function, but feel free to write additional test cases to make sure your code works. (Additional test cases are not requested, and will not be graded).

Remarks:

- You are not allowed to use code from online resources. Your submission will be tested against that, and will receive a 0, and a report to the Graduate Academic Integrity Board if it is detected.
- No additional libraries are allowed to be used, besides the ones already included in the source code shared with you.
- Makefiles are provided for both problems to build the code in LinuxLab.
- The programming, and testing will take some time. Start early.
- Feel free to use the provided source code for your implementation. You have to document your code.

Stevens Institute of Technology

Department of Computer Science

CS590: ALGORITHMS

Homework Assignment 2

Submitted by:
Hadia Hameed
CWID: 10440803

Yijia Tan
CWID: 10427079

Submitted to:
Prof. Iraklis Tsekourakis

October 10, 2018

1 Problem I

Design, and implement two algorithms to remove duplicates from an unsorted single Linked-List:

1.1 For the first algorithm, you can use an additional data structure.

Algorithm 1 Remove duplicates from a Linked List using an STL container.

```
procedure REMOVEDUPL1
    vector $v \leftarrow$ Linked list $myList$
    pointer $temp \leftarrow head$ of $myList$
    mergeSort($v$)
    $i \leftarrow 0$
    while $temp$ is not NULL do
        if $i < v.size$ and $v(i) = v(i+1)$ then
            $duplicateNode \leftarrow temp.next$
            $temp.next \leftarrow temp.next.next$
            delete $duplicateNode$
            $i \leftarrow i + 1$
        else
            $temp.data \leftarrow v(i)$
            $temp \leftarrow temp.next$
            $i \leftarrow i + 1$
        if $i > v.size$ then
            $temp \leftarrow NULL$
```

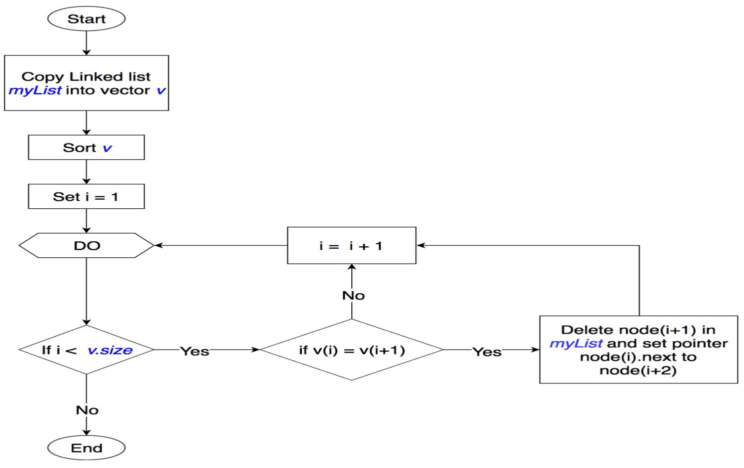
Start
Copy Linked list myList into vector v
Sort v
Set i = 1
DO
i = i + 1
No
If i < v.size
Yes
if v(i) = v(i+1)
Yes
Delete node(i+1) in myList and set pointer node(i).next to node(i+2)
No
End

Figure 1: Code flow for removeDupl_1

1.2 For the second algorithm, no additional data structure can be used.

Algorithm 2 Remove duplicates from a Linked List without using any other data structure.

```
procedure REMOVEDUPL2
    pointer $temp1 \leftarrow head$ of $myList$
    while $temp1$ is not NULL and $temp1.next$ is not NULL do
        pointer $temp2 \leftarrow temp1$
        while $temp2.next$ is not NULL do
            if $temp1.data = temp2.next.data$ then
                pointer $duplicateNode \leftarrow temp2.next$
                $temp2.next \leftarrow temp2.next.next$
                delete $duplicateNode$
            else
                $temp2 \leftarrow temp2.next$
        $temp1 \leftarrow temp1.next$
```

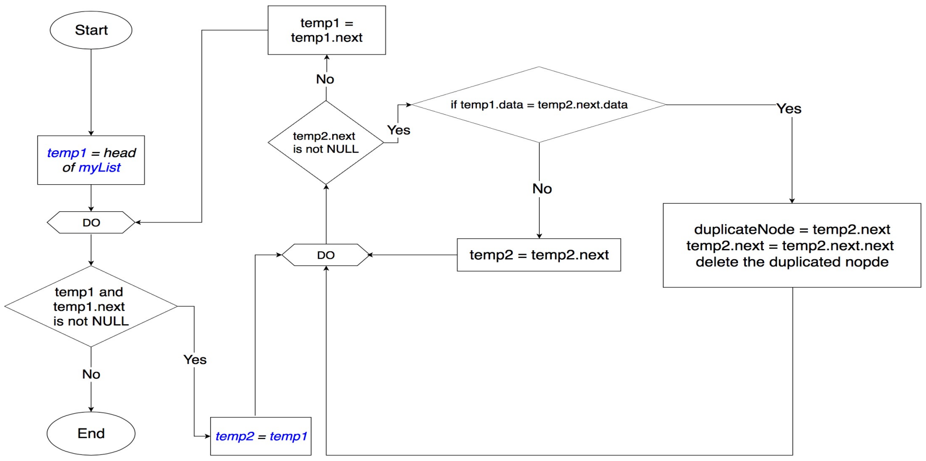
Start
temp1 = head of myList
DO
temp1 and temp1.next is not NULL
No
End
Yes
temp2 = temp1
DO
temp2.next is not NULL
No
temp1 = temp1.next
Yes
if temp1.data = temp2.next.data
No
temp2 = temp2.next
Yes
duplicateNode = temp2.next
temp2.next = temp2.next.next
delete the duplicated nopde

Figure 2: Code flow for removeDupl_2

2 Problem II

Design, and implement a C++ class that implements the queue data structure, using two stacks:

```
stack<int> stack1, stack2;
```

```
void EnQueue(int data){
        stack1.push(data);
    }

    int DeQueue(){
        if(stack1.empty()){
            cout<<"Stack is empty."<<endl;
            return false;
        }
    /*Transfer elements from one stack to another so the first in
      comes on top of the second stack*/
        while(!stack1.empty()){
            stack2.push(stack1.top());
            stack1.pop();
        }

        int j = stack2.top();
        stack2.pop();

        /*Place the elements back in stack1 from stack2*/
        while(!stack2.empty()){
            stack1.push(stack2.top());
            stack2.pop();
        }
        return j;
    }
```