**Sandesh Ghimire**

**CSCI 301 section 3**

**Computer Science 2**

**Project 4**

**28th September**

**Introduction**

An ordered list is a sequence of elements, all of the same type, that is ordered by the elements ' values. One way to represent an unordered list is in a doubly linked list. This project contains three files, " *List.h* “, " *List.cpp* " and " *main.cpp* “. The program initializes the doubly linked list. The user enters a string with ‘#’ which represents to take out the last alphabet of that string.

**Data Structures:**

The data structures used in this project is pointers. The pointers are used in order to initialize the linked list and access the contents of the doubly linked list. We used pointers to point the data and address of the list from the both ends.

We used linked list as a data structure. Linked list uses pointers to point the data and addresses in the list which makes it easier for us to insert and remove items.

We also used struct data structures which helps us to build Nodes. Node plays the major part of the list. The node holds all the address, data and links of linked list.

**Functions:**

The program contains a class named List in the header file " *List.h* “. The class list contains many public functions that are used to manipulate the linked list according to the user's decision the various function to manipulate the linked list are as follows:

*List() {first = NULL;*

*Last=NULL;}*: This is the constructor for the class List.

*~List():* This is the destructor for the class List.

*void append (Item empty):* This function inserts the value into the last position of list.

*void remove\_last (Item target):* This function removes the last value from the List.

*bool remove():* This function is used to find out if the list is empty or not.

*void Display:* This function prints the existing list.

*friend std::ostream& operator << (std::ostream& out\_s,const list& l):* This is a friend function to overload the “<<” operator.

**The Main Program:**

The main program utilizes the class and print outs the message to ask input.

At first, the main program asks for the input. Each characters of the input string is taken out using for loop. Then the function checks if the character is ‘#’ or not. If the character is ‘#’ we call the function *remove\_last()* to delete the last item. If the function does not contains ‘#’ then we used *append()* function to insert the data at last. At last the program displays the operated list using *Display()* function.

**Code**

**List.cpp**

#include <iostream>

#include <cassert>

#include "list.h"

using namespace std;

list::~list() //distructor

{

Node\*temp; //initalizing the temp pointer

while(first != NULL){ //using precondition loop

temp=first;

first=first->next;

delete temp;

}

}

void list::Display(){

Node \* temp; //declare the variable

if (first==NULL){cout<<"The list is empty.";} //pre condition

else{

int i=0;

temp=first;//give value to temp

while (temp != NULL){//post condition

i++;

cout<<temp->data;//output the data inside pointer

temp=temp->next;//points to next pointer

}

}

}

void list::append(const Item& entry){//function to insert

Node\*temp;

temp= new Node;//new node is declared

Node\* prev;

prev=first;//give first value to temp

if (first==NULL){//checking if the first value is NULL or not

first=get\_node(entry,last,first);//insert value

}else{

while(prev->next != NULL){//loop to get to the last pointer

prev=prev->next;//giving next address to prev pointer

}

temp->data=entry;//inserting data

prev->next=temp;//pointing next value

temp->next=last;

temp->back=prev;//pointing back

}

}

ostream& operator<<(ostream& out\_s, const list& b){ //overloading the << operator

list::Node\*n;

n= b.first;

cout<<n->data<<"";//displaying if null

while (n->next != NULL)

{

n=n->next;

out\_s<<n->data<<"";//displaying if data are present

}

return out\_s;

}

void list::remove\_last(const Item& target)//to remove

{

Node \* prev; //declare pointers

Node \* temp;

prev=first;

if(first->data==target){//pre condition

first=first->next;

delete prev;

}

else{

while (prev->next && prev->next->next != NULL)//loop started to check the last item

{

prev = prev->next;//pointing to next address

}

delete prev->next;//deleting the prev address

prev->next = NULL;//pointing to NULL

}

}

bool list::empty(){ //check if list is empty or not

if(first==NULL&&last==NULL){//pre condition

return true;

}

else{//post condition

return false;

}

}

list::Node\* list::get\_node(const Item value, Node\* link, Node\* backward){

Node\*temp;

temp= new Node;//decclearing new node

temp->data=value;//giving data

temp->next=link;//getting the value of particular node

temp->back=backward;//pointing backwards

return temp;//returning the value of temp

}

**Main.cpp**

#include <iostream>

#include <cassert>

#include "list.h"

#include "list.cpp"

using namespace std;

int main(){ //The main function

list l;

char choice;

string Inp;

cout<<"Enter a line of characters: "<<endl;//displaying message

cin>>Inp;//input in a veriable

cout<<"The previous value is "<<endl;//printing the original string

cout<<Inp<<endl;

for (int i=0;i<Inp.length();i++)//starting the loop

{

if(Inp[i]=='#'){//if value is # then

l.remove\_last(Inp[i-1]);}//remove the value

else{

l.append(Inp[i]);//otherwise insert value to the list

}

}

cout<<"The after operation is "<<endl;

l.Display();//display the whole list

}

**List.h**

#include <iostream>

#ifndef LIST\_H

#define LIST\_H

using namespace std;

class list{

public://public functions of the class

typedef char Item; //defining the Item using int

//Item first;

list(){first=NULL;

last=NULL;}; //constructor

~list();//Destructor

void append (const Item& entry);//to insert the data in the list

void remove\_last(const Item& entry);//to remove the data from the list

//Member functions

bool empty();//to check if list is empty

void Display();//prints out the contents of list

friend std::ostream& operator << (std::ostream& out\_s,const list& list1); //this is a friend function

private://private functions of class

struct Node //Making a Node for pointing the data of list

{

Item data;//data

Node\* next;//pointer

Node\* back;

};

Node\* first;//It points first data of list

Node\* last;//It points last of the list

Node\* get\_node(Item Empty, Node \*link, Node\* Backward);//to get the data in list

};

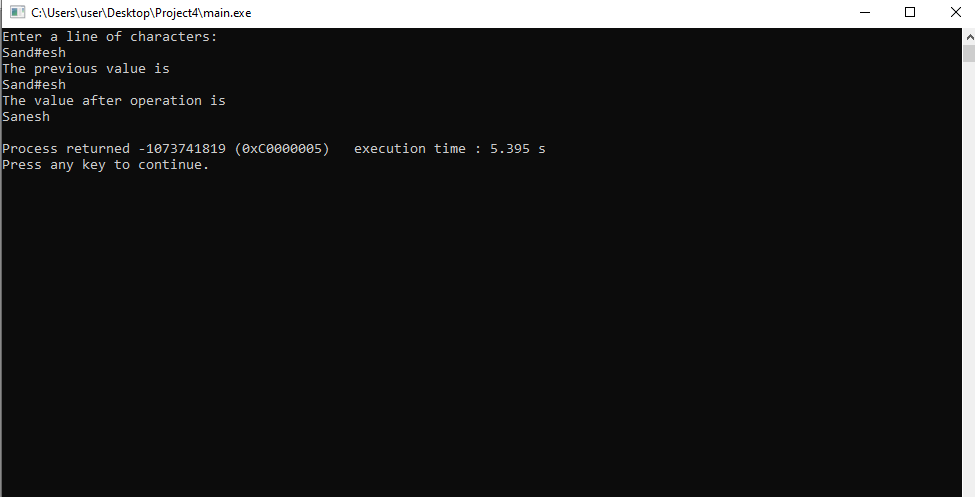
#endif // list\_H

**User Document**

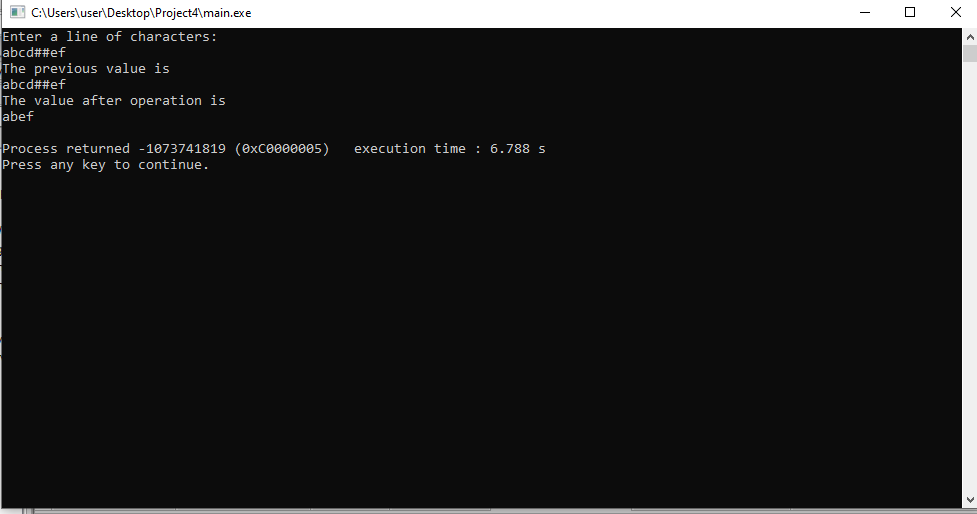
The program can be used by entering a string that contains that contains ‘#’. # means remove the last data of the list. So, whenever we see the character has ‘#’ we take out the last character of the doubly linked list. To compile the program user can open the executable file. As soon as the .exe file opens, the user will be asked to enter the desired action to be taken. The user will be able to see the result after the program is executed.

**Testing**

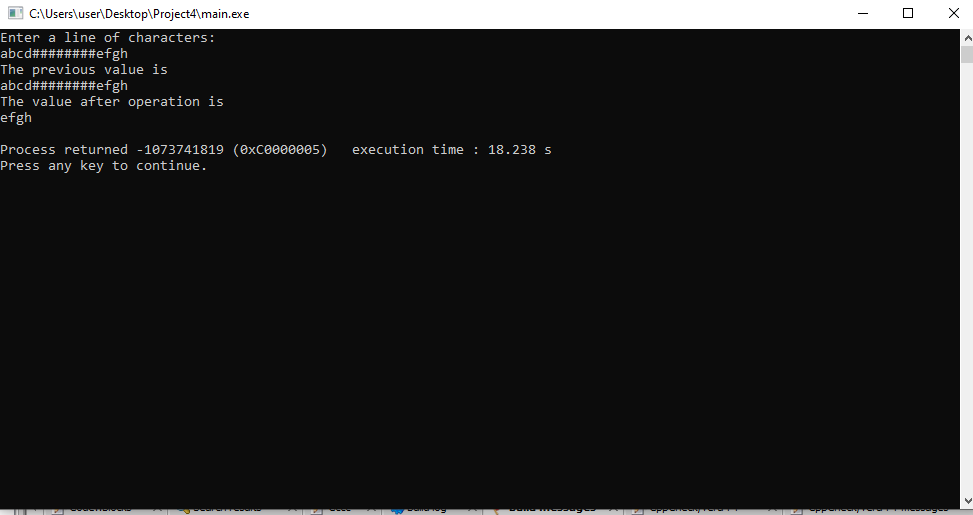
TEST1



TEST2



TEST3



**Summary**

In this project, we have implemented a program that uses doubly linked list. The program reads the string data from the user. I used three files to keep my codes organized and easy to access. The problems and their solutions involving doubly linked list that we learnt in class helped a lot to solve the given question. This problem can be solved more easily if we used stack (ADT) which follows last in first out method.