DATA STRUCTURES AND ALGORITHMS

HASSAAN AKBAR CHEEMA

REGN NO. 174351

LAB 6

// ConsoleApplication1.cpp : Defines the entry point for the console application.

//

#include<iostream>

using namespace std;

struct node

{

int data;

node \*next;

};

class list

{

int sum = 0;

private:

node \*head, \*tail;

public:

list()

{

head = NULL;

tail = NULL;

}

void apendnode(int value)

{

node \*temp = new node;

temp->data = value;

temp->next = NULL;

if (head == NULL)

{

head = temp;

tail = temp;

temp = NULL;

}

else

{

tail->next = temp;

tail = temp;

}

}

void display()

{

if (head != NULL) {

node \*temp = new node;

temp = head;

while (temp != NULL)

{

cout << temp->data << "\t";

temp = temp->next;

}

}

else

cout << "list is empty" << endl;

}

int add() {

node\* ptr = head;

if (head != NULL) {

do {

sum += ptr->data;

ptr = ptr->next;

} while (ptr != NULL);

}

return sum;

}

int search(int x) {

node \* ptr = head;

int i = 1;

if (head != NULL) {

while (ptr->data != x && ptr->next != NULL) {

ptr = ptr->next;

i++;

}

if (ptr->data == x && ptr->next == NULL) {

cout << "Element " << ptr->data << " is found at position: " << i << endl;

return ptr->data;

}

else if (ptr->data == x) {

cout << "Element " << ptr->data << " is found at position: " << i << endl;

return ptr->data;

}

else if (ptr->next == NULL && ptr->data != x) {

cout << "NO SUCH ELEMENT FOUND" << endl;

return NULL;

}

}

else

display();

}

void destroy\_list() {

if (head != NULL) {

node \* ptr = head;

while (ptr->next != NULL) {

head = ptr->next;

delete ptr;

ptr = head;

}

head = NULL;

delete ptr;

ptr = NULL;

}

else

display();

}

void insertatbeg(int x) { // insert node at start

node\* newnode = new node;

newnode->next = head;

newnode->data = x;

head = newnode;

}

void insertatend(int x) { // insert node at end

node\* newnode = new node;

newnode->data = x;

node\* ptr;

if (head != NULL) {

ptr = head;

while (ptr->next != NULL) {

ptr = ptr->next;

}

ptr->next = newnode;

newnode->next = NULL;

}

else

insertatbeg(x);

}

void insertnode(int x, int y) { // insert node at given

node\* newnode = new node; // location

if (x <= numofnodes()) {

int counter = 2;

node \*ptr, \*ptr2;

ptr = head;

ptr2 = ptr->next;

while (counter < x) {

ptr = ptr->next;

ptr2 = ptr2->next;

counter++;

}

if (x == 1) {

newnode->next = ptr;

newnode->data = y;

head = newnode;

}

else {

newnode->next = ptr2;

ptr->next = newnode;

newnode->data = y;

}

}

else

cout << "LIMIT EXCEEDED, ENTER AT LOCATIONS <= " << numofnodes() << endl;

}

int numofnodes() {

node\* ptr;

if (head != NULL) {

ptr = head;

int counter = 2;

while (ptr->next != NULL) {

ptr = ptr->next;

counter++;

}

return counter;

}

else

return 1;

}

void deleteatstart() { // delete the first node

node\* ptr;

if (head != NULL) {

ptr = head;

head = ptr->next;

ptr->next = NULL;

ptr = head;

}

else

display();

}

void deleteatend() { // delete the last node

node\* ptr;

if (head != NULL) {

ptr = head;

if (ptr->next == NULL) {

head = NULL;

delete ptr;

ptr = NULL;

}

else {

while ((ptr->next)->next != NULL) {

ptr = ptr->next;

}

ptr->next = NULL;

}

}

else

display();

}

void deletenode(int x) // delete node at given location

{

if (head != NULL) {

if (x < numofnodes()) {

int counter = 2;

node \*ptr, \*ptr2;

ptr = head;

ptr2 = ptr->next;

while (counter < x) {

ptr = ptr->next;

ptr2 = ptr2->next;

counter++;

}

if (x == 1) {

head = ptr->next;

ptr->next = NULL;

}

else {

ptr->next = ptr2->next;

ptr2->next = NULL;

}

}

else

cout << "LIMIT EXCEEDED, DELETE AT LOCATIONS < " << numofnodes() << endl;

}

else

display();

}

void delete\_by\_value(int x) {

node \* ptr = head;

int i = 1;

while (ptr->data != x && ptr->next != NULL) {

ptr = ptr->next;

i++;

}

if (ptr->data == x && ptr->next == NULL) {

cout << "Element " << ptr->data << " is found at position: " << i << endl;

deletenode(i);

}

else if (ptr->data == x) {

cout << "Element " << ptr->data << " is found at position: " << i << endl;

deletenode(i);

}

else if (ptr->next == NULL && ptr->data != x) {

cout << "NO SUCH ELEMENT FOUND" << endl;

}

}

bool isempty() {

if (head == NULL)

return true;

else

return false;

}

};

void main()

{

int choice, value, SUM = 0, select = 0, program = 1;

list li = list();

cout << " WELCOME TO THE LINK LIST " << endl << endl;

while (program == 1) {

cout << "1 TO APPEND NODES " << endl;

cout << "2 TO DISPLAY, DESTROY OR SEARCH " << endl;

cout << "3 TO INSERT OR DELETE NODE " << endl;

cout << "SELECT: ";

cin >> select;

if (select == 1) {

/\*cout << "1 TO APPEND" << endl;

cin >> choice;\*/

choice = 1;

if (choice == 1) {

cout << "HOW MANY NODES YOU WANT TO APPEND: ";

cin >> value;

}

for (int i = 0; i < value; i++) {

int num = 0;

cout << "ENTER THE VALUE: ";

cin >> num;

li.apendnode(num);

}

}

if (select == 2) {

if (li.isempty())

cout << "LIST IS EMPTY" << endl;

else {

int startel = 1;

while (startel == 1) {

cout << "1 TO DISPLAY THE LINK LIST >" << endl;

cout << "2 DISPLAY THE ADDED UP VALUES OF LINK LIST >" << endl;

cout << "3 TO SEARCH ANY ELEMENT >" << endl;

cout << "4 TO DESTROY THE LIST >" << endl;

cout << "CHOICE: " ;

cin >> choice;

if (choice == 1) {

li.display();

cout << endl;

}

if (choice == 2) {

SUM = li.add();

cout << "SUM OF ALL THE NODES: " << SUM << endl;

}

if (choice == 3) {

int a;

cout << "WHICH ELEMENT YOU WANT TO SEARCH: ";

cin >> a;

li.search(a);

}

if (choice == 4) {

li.destroy\_list();

}

cout << endl << "To display, destroy or search the linked list press 1 and to continue press 2: ";

cin >> startel;

}

}

}

if (select == 3) {

int start = 1;

while (start == 1) {

cout << endl << "What you want to do with the list:" << endl;

cout << "Press 1 to insert node at start:" << endl;

cout << "Press 2 to insert node at end:" << endl;

cout << "Press 3 to insert node at a particular position:" << endl;

cout << "Press 4 to delete node at start:" << endl;

cout << "Press 5 to delete node at end:" << endl;

cout << "Press 6 to delete node at a particular position:" << endl;

cout << "Press 7 to delete node with a particular value:" << endl;

cout << "Choice: ";

cin >> choice;

int newvalue, location;

if (choice == 1) {

cout << "What value you want to store in the new node: ";

cin >> newvalue;

li.insertatbeg(newvalue);

li.display();

cout << endl;

}

if (choice == 2) {

cout << "What value you want to store in the new node: ";

cin >> newvalue;

li.insertatend(newvalue);

li.display();

cout << endl;

}

if (choice == 3) {

cout << "At what location you want to insert the new node: ";

cin >> location;

cout << "What value you want to store in the new node: ";

cin >> newvalue;

li.insertnode(location, newvalue);

li.display();

cout << endl;

}

if (choice == 4) {

li.deleteatstart();

li.display();

cout << endl;

}

if (choice == 5) {

li.deleteatend();

li.display();

cout << endl;

}

if (choice == 6) {

int location;

cout << "At what location you want to delete the node: ";

cin >> location;

li.deletenode(location);

li.display();

cout << endl;

}

if (choice == 7) {

int newvalue;

cout << "At what value you want to delete the node: ";

cin >> newvalue;

li.delete\_by\_value(newvalue);

li.display();

cout << endl;

}

cout << endl << "To alter the linked list press 1 and to end the program press 2: ";

cin >> start;

}

}

cout << endl << "To continue press 1 and to end the program press 2: ";

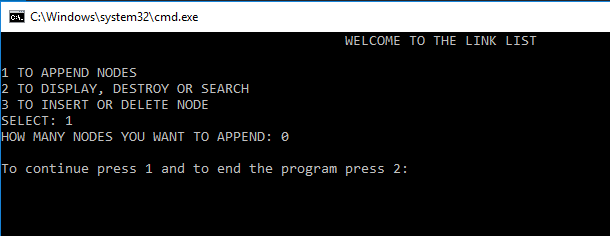
cin >> program;

}

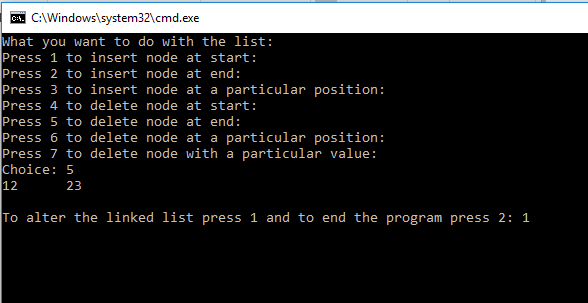
}

L0: and test all the above functions by using it. Following functions should work:

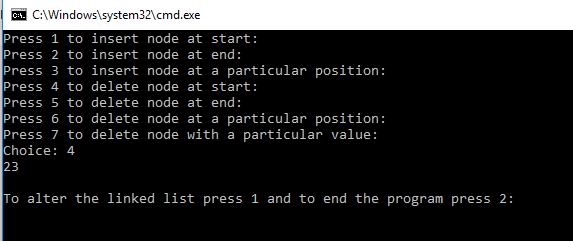
**Create L0 with no nodes at all.**



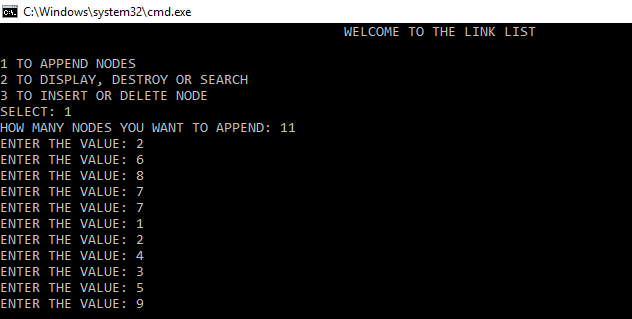
**Try to delete the last node.**



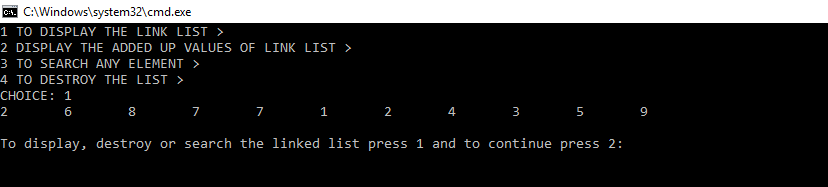
**Try to delete the node at the start.**



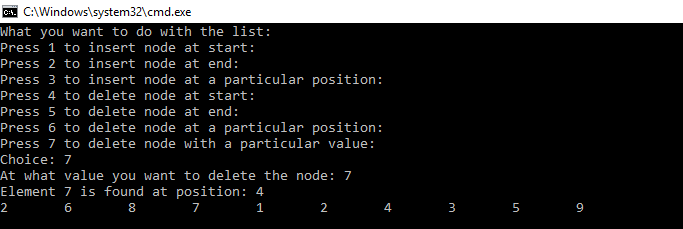
**Append 2, Append 6, Append 8, Append 7, Append 7, Append 1, Append 2, Append 4, Append 3, Append 5, Append 9.**



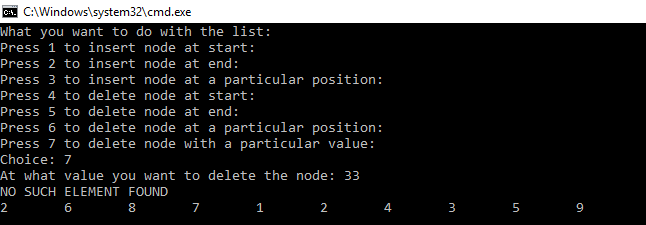
**Display the list.**



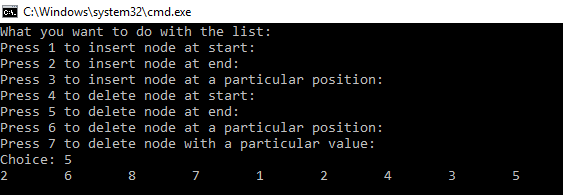
**Try to delete the node with value 7**



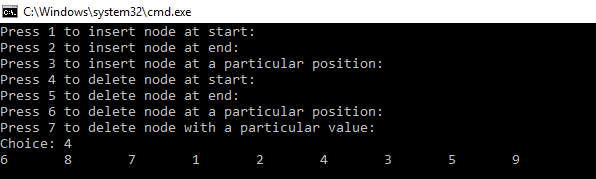
**Try to delete the node with value 33**



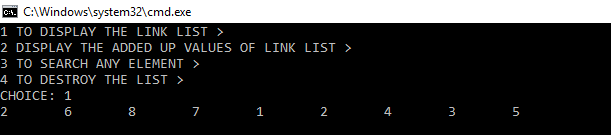
**Try to delete the last node**



**Try to delete the first node**



**Display the list.**



Task 2: Create a new linked list L1 and do the following:

**L1: 5, 10,15,20,25,30,35,40**

**Create L1 by using appendNode()**

**Create L2 and L3 in which you remove values from front from L1 and add the deleted values from L1 to L2 and L3 alternately. For example if**

**L1: 5, 10, 15, 20, 25, 30, 35, 40 then**

**L2: 5, 15, 25, 35**

**L3:10, 20, 30, 40**

**Check your code by displaying all the three Lists (L1 to L3)**

#include<iostream>

using namespace std;

struct node

{

int data;

node \*next;

};

class list

{

public:

node \*head, \*tail;

public:

list()

{

head = NULL;

tail = NULL;

}

void apendnode(int value)

{

node \*temp = new node;

temp->data = value;

temp->next = NULL;

if (head == NULL)

{

head = temp;

tail = temp;

temp = NULL;

}

else

{

tail->next = temp;

tail = temp;

}

}

void display()

{

if (head != NULL) {

node \*temp = new node;

temp = head;

while (temp != NULL)

{

cout << temp->data << "\t";

temp = temp->next;

}

}

else

cout << "list is empty" << endl;

}

void deleteatstart() { // delete the first node

node\* ptr;

if (head != NULL) {

ptr = head;

head = ptr->next;

ptr->next = NULL;

ptr = head;

}

else

display();

}

};

void main() {

list li1 = list();

list li2 = list();

list li3 = list();

li1.apendnode(5);

li1.apendnode(10);

li1.apendnode(15);

li1.apendnode(20);

li1.apendnode(25);

li1.apendnode(30);

li1.apendnode(35);

li1.apendnode(40);

cout << "li1 = ";

li1.display();

cout << endl;

for (int x = 1; x < 9; x++) {

if (x % 2 == 1) {

li2.apendnode(li1.head->data);

li1.deleteatstart();

}

else if(x % 2 == 0){

li3.apendnode(li1.head->data);

li1.deleteatstart();

}

}

cout << "li2 = ";

li2.display();

cout << endl;

cout << "li3 = ";

li3.display();

cout << endl;

}

