**Code**

#include<iostream>

using namespace std;

class node {

public:

int val;

node\* next;

node(int data) {

val = data;

next = NULL;

}

};

//insert in the start

node\* inshead(node\* &head, int val) {

node\* new\_node = new node(val);

new\_node->next = head;

head = new\_node;

return head;

}

//insert in the end

node\* insertAtEnd(node\* &head, int val)

{

node\* new\_node= new node(val);

if(head==NULL){

head=new\_node;

}

node\* temp=head;

while(temp->next!=NULL){

temp=temp->next;

}

temp->next=new\_node;

return head;

}

//insert at N position

node\* insertatN(node\* head, int val, int pos){

if(pos==0){

return inshead(head, val);

}

node\* new\_node=new node(val);

node\* temp=head;

for(int i=0; i<pos-1 && temp!=NULL; ++i){

temp=temp->next;

}

if(temp==NULL)

{

cout<<"insertion at N is not possible "<<endl;

delete new\_node;

}

else{

new\_node->next= temp->next;

temp->next= new\_node;

}

return head;

}

//delete from start

node\* del\_start(node\* &head){

if(head==NULL)

{

cout<<"deletion not possible "<<endl;

}

else {

node\* temp=head;

head= temp->next;

delete temp;

}

return head;

}

//delete from end

node\* del\_end(node\* &head) {

if (head == NULL) {

cout << "Deletion not possible. " << endl;

}

else if (head->next == NULL) {

delete head;

head = NULL;

}

else {

node\* temp = head;

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

temp = temp->next;

}

delete temp->next;

temp->next = NULL;

}

return head;

}

//deletion from Nth position

node\* del\_atN(node\* &head, int pos) {

if (head == NULL) {

cout << "Deletion not possible. List is empty." << endl;

}

else if (pos == 0) {

return del\_start(head);

}

else {

node\* temp = head;

for (int i = 0; i < pos - 1 && temp != NULL; ++i) {

temp = temp->next;

}

if (temp == NULL || temp->next == NULL) {

cout << "Deletion at position " << pos << " is not possible." << endl;

}

else {

node\* toDelete = temp->next;

temp->next = temp->next->next;

delete toDelete;

}

}

return head;

}

//search any value

node\* searchValue(node\* head, int val) {

node\* temp = head;

int pos = 0;

while (temp != NULL) {

if (temp->val == val) {

cout << "Value " << val << " found at position " << pos << "." << endl;

return temp;

}

temp = temp->next;

pos++;

}

cout << "Value " << val << " not found in the list." << endl;

return NULL;

}

//to update values

void updateValue(node\* head, int oldVal, int newVal) {

node\* temp = head;

while (temp != NULL) {

if (temp->val == oldVal) {

temp->val = newVal;

cout << "Value " << oldVal << " updated to " << newVal << "." << endl;

return;

}

temp = temp->next;

}

cout << "Value " << oldVal << " not found in the list. Update failed." << endl;

}

//function t display elements

void displayList(node\* head) {

node\* temp = head;

while (temp != NULL) {

cout << temp->val << " ";

temp = temp->next;

}

}

int main() {

int val;

node\* a = new node(5);

a->next = new node(6);

a->next->next = new node(7);

cout<<"current list: "<<endl;

displayList(a);

cout<<endl;

//inserting the value at the start

a = inshead(a, 2);

cout << "After insertion at start "<<endl;

displayList(a);

a= insertAtEnd(a,9);

cout<<endl;

cout<<"insertion at the end: "<<endl;

displayList(a);

//insertion at any N position

a= insertatN(a, 0,3);

cout<<endl;

cout<<"insert at any N position "<<endl;

displayList(a);

cout<<endl;

//deletion from start

a= del\_start(a);

cout<<endl;

cout<<"after deletion from start "<<endl;

displayList(a);

//deletion from end

a= del\_end(a);

cout<<endl;

cout<<"after deletion from end. "<<endl;

displayList(a);

//deletion from Nth position

a= del\_atN(a, 1);

cout<<endl;

cout<<"after deletion from nth position "<<endl;

displayList(a);

cout<<endl;

//search value

node\* result = searchValue(a, 3);

cout<<endl;

//update value

updateValue(a, 0,10);

cout<<"updated list: "<<endl;

displayList(a);

cout<<endl;

return 0;

}

**Output**

