**Write Doubly Link List code with functions perform on it.**

**Code:** #include<iostream> using namespace std;

class Node{ public: int data;

Node\* prev;

Node\* next;

//constructor Node(int d){ this -> data = d; this -> prev = NULL; this -> next = NULL;

}

//desturctor ~Node(){ int val = this ->data; if(next != NULL){ delete next; next = NULL;

}

cout<< "Memory free for Node with data"<<endl;

}

};

//Data in the doubly linked list

void printLIST(Node\* head){ Node\* temp=head; cout << " [ ";

while (temp!=NULL){ cout<< temp -> data<<" "; temp = temp -> next;

}

cout << "] " << endl;

}

//Length of the doubly linlked List int getLength(Node\* head){

int len = 0;

Node\* temp=head; while (temp!=NULL){

len++; temp = temp -> next;

} return len; }

void insertAtheadLL(Node\* &tail,Node\* &head,int d){

//Empty List if(head == NULL){ Node\* temp = new Node(d); head = temp; tail = head;

}

else{

Node\* temp = new Node(d);

temp -> next =head; head -> prev = temp; head = temp;

}

}

void insertAttail(Node\* &tail,int d){ Node\* temp= new Node(d); tail -> next = temp; temp -> prev = tail; tail = temp;

}

void insertAtposition(Node\* &tail,Node\* &head,int position,int d){ if(position==1){ insertAtheadLL(tail,head,d); return;

}

if(position>((getLength(head))+1)){ position= (getLength(head))+1; insertAtposition(tail,head,position,d); return;

}

Node \* temp = head; int count = 1; while(count < position-1){ temp = temp-> next; count++;

}

if(temp -> next == NULL){

insertAttail(tail,d); return;

}

//create nodeTo Insert

Node\* nodeToinsert = new Node(d);

nodeToinsert -> next = temp -> next;

temp -> next -> prev = nodeToinsert;

temp -> next = nodeToinsert;

nodeToinsert -> prev = temp;

} void delatbegin(Node\* &head)

{

head = head->next;

}

void delatend(Node\* &tail)

{

tail = tail->prev; tail->next = NULL;

}

void deleteNode(int position,Node\* &head){

if(position == 1)//for deleting the first Node

{

Node\* temp = head; temp -> next -> prev = NULL; head = temp -> next; //memory free node temp -> next = NULL; delete temp;

}

else{//delete other node with last Node

Node\* curr = head;

Node\* prev = NULL;

int count = 1; while (count < position)

{ prev = curr; curr = curr -> next; count++;

/\* code \*/

}

curr -> prev = NULL; prev -> next = curr -> next; curr -> next = NULL; delete curr;

}

} //searching int search(Node\* &head,int data){

Node\* trav = head;

bool flag = false; int count = 0; while(trav != NULL && flag == false){

count++;

if(trav -> data == data){

flag = true; return count;

break;

}

else{

trav = trav -> next;

}

}

if(flag == true){ cout << "Element Found!" << endl;

}

else{

cout << "Element Not Found!" << endl;

}

return 0;

}

//reverse void reverseList(Node\*\* head){

Node\* prev=NULL,\*cur=\*head,\*tmp; while(cur!=NULL){ tmp=cur->next; cur->next=prev; prev=cur;

cur=tmp;

}

\*head=prev;

}

int main(){

Node\* head = NULL;

Node\* tail = NULL;

insertAtheadLL(tail,head,11); printLIST(head);

cout<<"\nLength of the Node:"<< getLength(head)<<endl;

insertAtheadLL(tail,head,17); printLIST(head);

cout<<"\nLength of the Node:"<< getLength(head)<<endl;

insertAtheadLL(tail,head,8); printLIST(head);

cout<<"\nLength of the Node:"<< getLength(head)<<endl;

insertAtheadLL(tail,head,1); printLIST(head);

cout<<"\nLength of the Node:"<< getLength(head)<<endl; insertAttail(tail,25); printLIST(head);

insertAtheadLL(tail,head,88); printLIST(head);

cout<<"\nLength of the Node:"<< getLength(head)<<endl;

insertAtheadLL(tail,head,127); printLIST(head);

cout<<"\nLength of the Node:"<< getLength(head)<<endl;

insertAtheadLL(tail,head,83); printLIST(head);

cout<<"\nLength of the Node:"<< getLength(head)<<endl;

cout<<"\n Reversed Linked list: "; reverseList(&head);

printLIST(head);

cout<<"\nRemove Element at end :"<<endl; delatend(tail); printLIST(head);

cout<<"\nLength of the Node:"<< getLength(head)<<endl; cout<<"\nRemove Element at begin :"<<endl; delatbegin(head); printLIST(head);

cout<<"\nLength of the Node:"<< getLength(head)<<endl;

insertAtposition(tail,head,8,77); printLIST(head);

cout<<"\nLength of the Node:"<< getLength(head)<<endl;

int p;

cout << "\nEnter Position you want remove from the Linked List: ";

cin>>p;

deleteNode(p,head);

cout<<"\nRemove Node At position " << p <<":"; printLIST(head);

cout<<"\nSearching Element in the Linked list: ";

int n; cin >> n;

cout <<"\nLocation of the Element is "<<search(head,n);

cout<<"\nLength of the Node:"<< getLength(head)<<endl;

return 0;

}

