1. Doubly linked list(creation ,insertion at beg ,insertion at end)

//#include<stdio.h>

#include<iostream>

#include<stdlib.h>

using namespace std;

struct node{

struct node \*prev;

int data;

struct node \*next;

};

//traversing elements

void traverse(struct node\* head)

{

struct node\* ptr=head;

while(ptr!=NULL)

{

cout<<ptr->data<<endl;

ptr=ptr->next;

}

}

//insert elements function

struct node\* addToEmpty(struct node\* head,int data)

{

struct node\* temp =(struct node \*)malloc(sizeof(struct node));

temp->prev=NULL;

temp->data=data;

temp->next=NULL;

head=temp;

return head;

}

//insert at beg

struct node\* insert\_beg(struct node\* head,int data)

{

struct node\* temp =(struct node \*)malloc(sizeof(struct node));

temp->prev=NULL;

temp->data=data;

temp->next=head;

head->prev=temp;

head=temp;

return head;

}

//insert at end

void insert\_end(struct node \*head,int data)

{

struct node \*ptr, \*temp;

ptr=head;

temp=(struct node \*)malloc(sizeof(struct node));

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

temp->data=data;

temp->prev=ptr;

temp->next=NULL;

ptr->next=temp;

}

int main()

{

// creation and insertion

struct node \*head=NULL;

/\*head=(struct node \*)malloc(sizeof(struct node));

head->prev=NULL;

head->data=45;

head->next=NULL;

\*/

head=addToEmpty(head,45);

// traversing

traverse(head);

// insert data at beg

head=insert\_beg(head,46);

//traverseafter insert at beg

cout<<"traverse elemnts after inserting data at beg"<<endl;

traverse(head);

// insert at end

insert\_end(head,54);

//traverse after insert dataa at end

cout<<"traverse elemnts after inserting data at end"<<endl;

traverse(head);

return 0;

}

// Insertion in doubly linked list using new

#include <iostream>

using namespace std;

struct node {

struct node \*prev;

int data;

struct node \*next;

};

// Traversing elements

void traverse(struct node\* head) {

struct node\* ptr = head;

while(ptr != NULL) {

cout << ptr->data << endl;

ptr = ptr->next;

}

}

// Insert elements function

struct node\* addToEmpty(struct node\* head, int data) {

struct node\* temp = new node;

temp->prev = NULL;

temp->data = data;

temp->next = NULL;

head = temp;

return head;

}

// Insert at beginning

struct node\* insert\_beg(struct node\* head, int data) {

struct node\* temp = new node;

temp->prev = NULL;

temp->data = data;

temp->next = head;

if (head != NULL)

head->prev = temp;

head = temp;

return head;

}

// Insert at end

void insert\_end(struct node\* head, int data) {

struct node\* ptr = head;

struct node\* temp = new node;

temp->data = data;

temp->next = NULL;

while(ptr->next != NULL) {

ptr = ptr->next;

}

temp->prev = ptr;

ptr->next = temp;

}

int main() {

// Creation and insertion

struct node\* head = NULL;

head = addToEmpty(head, 45);

// Traversing

traverse(head);

// Insert data at beginning

head = insert\_beg(head, 46);

// Traverse after inserting data at beginning

cout << "Traverse elements after inserting data at beginning" << endl;

traverse(head);

// Insert at end

insert\_end(head, 54);

// Traverse after insert data at end

cout << "Traverse elements after inserting data at end" << endl;

traverse(head);

return 0;

}

2) Deletion at beg and from end

//#include<stdio.h>

#include<iostream>

#include<stdlib.h>

using namespace std;

struct node{

struct node \*prev;

int data;

struct node \*next;

};

//traversing elements

void traverse(struct node\* head)

{

struct node\* ptr=head;

while(ptr!=NULL)

{

cout<<ptr->data<<endl;

ptr=ptr->next;

}

}

//insert elements function

struct node\* addToEmpty(struct node\* head,int data)

{

struct node\* temp =(struct node \*)malloc(sizeof(struct node));

temp->prev=NULL;

temp->data=data;

temp->next=NULL;

head=temp;

return head;

}

//insert at beg

struct node\* insert\_beg(struct node\* head,int data)

{

struct node\* temp =(struct node \*)malloc(sizeof(struct node));

temp->prev=NULL;

temp->data=data;

temp->next=head;

head->prev=temp;

head=temp;

return head;

}

//insert at end

void insert\_end(struct node \*head,int data)

{

struct node \*ptr, \*temp;

ptr=head;

temp=(struct node \*)malloc(sizeof(struct node));

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

temp->data=data;

temp->prev=ptr;

temp->next=NULL;

ptr->next=temp;

}

struct node\* delete\_beg(struct node\* head)

{

head=head->next;

head->prev=NULL;

return head;

}

struct node\* delete\_last(struct node\* head)

{

struct node\* temp1=head;

struct node\* temp2=head;

while(temp1->next!=NULL)

{

temp2=temp1;

temp1=temp1->next;

}

temp2->next=NULL;

//temp1->prev=NULL;

free(temp1);

return head ;

}

int main()

{

// creation and insertion

struct node \*head=NULL;

/\*head=(struct node \*)malloc(sizeof(struct node));

head->prev=NULL;

head->data=45;

head->next=NULL;

\*/

head=addToEmpty(head,45);

// traversing

traverse(head);

// insert data at beg

head=insert\_beg(head,46);

//traverseafter insert at beg

cout<<"traverse elemnts after inserting data at beg"<<endl;

traverse(head);

// insert at end

insert\_end(head,54);

//traverse after insert dataa at end

cout<<"traverse elemnts after inserting data at end"<<endl;

traverse(head);

//delete first node

head=delete\_beg(head);

//traverse delete first node

cout<<"traverse elemnts after deleting data from beg "<<endl;

traverse(head);

//delete last node

head=delete\_last(head);

//traverse delete last node

cout<<"traverse elemnts after deleting last node "<<endl;

traverse(head);

return 0;

}