**ASSIGNMENT-6**

Q1

#include <iostream>

using namespace std;

struct node{

int data;

struct node\*next;

};

void traversal(struct node\*p,struct node\*ptr){

while(ptr->next!=p){

cout<<ptr->data<<" ";

ptr=ptr->next;

}

cout<<ptr->data;

}

struct node\* insertbeg( struct node\*head,int val){

struct node\*ptr;

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

struct node\*p;

ptr->data=val;

p=head->next;

while(p->next!=head){

p=p->next;

}

p->next=ptr;

ptr->next=head;

head=ptr; // or can simply return ptr and store in head

return head;

}

struct node\* insertend( struct node\*head,int val){

struct node\*ptr;

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

ptr->data=val;

struct node\*p;

p=head;

while(p->next!=head){

p=p->next;

}

p->next=ptr;

ptr->next=head;

return head;

}

struct node\* insertval( struct node\*head,int val1,int val){

struct node\*ptr;

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

ptr->data=val;

struct node\*p;

p=head;

while(p!=NULL && p->data!=val1){

p=p->next;

}

ptr->next=p->next;

p->next=ptr;

return head;

}

struct node\* deletebeg( struct node\*head){

struct node \*ptr;

struct node \*p;

ptr=head;

p=head;

while(p->next!=head){

p=p->next;

}

head=head->next;

p->next=head;

free(ptr);

return head;

}

struct node\* deletend( struct node\*head){

struct node \*ptr;

struct node \*p;

ptr=head;

p=head->next;

while(p->next!=head){

p=p->next;

ptr=ptr->next;

}

ptr->next=head;

free(p);

return head;

}

struct node\* deletebet( struct node\*head,int val){

struct node\*ptr;

ptr=head;

struct node\* p;

p=head->next;

while(p!=NULL && p->data!=val){

p=p->next;

ptr=ptr->next;

}

ptr->next=p->next;

free(p);

return head;

}

int main()

{

struct node\*head;

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

struct node\*second;

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

struct node\*third;

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

struct node\*fourth;

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

head->data=7;

head->next=second;

second->data=11;

second->next=third;

third->data=66;

third->next=fourth;

fourth->data=23;

fourth->next=head;

head=deletebet(head,11);

traversal(head,head);

return 0;

}

(b)

#include <iostream>

using namespace std;

struct node{

int data;

struct node\*next;

struct node\*prev;

};

void search(struct node\*head ,int val){

struct node\*p;

p=head;

while(p!=NULL && p->data!=val){

p=p->next;

}

if(p==NULL){

cout<<"element not found";

}

else{

cout<<"element found";

}

}

void traversal(struct node\*head)

{

struct node\*ptr;

ptr=head;

while(ptr!=NULL){

cout<<ptr->data<<" ";

ptr=ptr->next;

}

}

struct node\* insertbeg( struct node\*head,int val){

struct node\*temp;

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

temp->data=val;

temp->prev=NULL;

temp->next=head;

head->prev=temp;

head=temp;

return head;

}

struct node\* insertend( struct node\*head,int val){

struct node\*temp;

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

temp->data=val;

temp->next=NULL;

temp->prev=NULL;

struct node\*p;

p=head;

while(p->next!=NULL){

p=p->next;

}

temp->next=NULL;

p->next=temp;

temp->prev=p;

return head;

}

struct node\*insertval( struct node\*head,int val1,int val2){

struct node\*temp;

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

temp->data=val1;

temp->next=NULL;

temp->prev=NULL;

struct node\*p;

p=head;

if(head==NULL){

cout<<"list is empty";

}

else{

while(p!=NULL && p->data!=val2){

p=p->next;

}

if(p==NULL){

cout<<"value not found";

}

else{

p->next->prev=temp;

temp->next=p->next;

p->next=temp;

temp->prev=p;

}

}

return head;

}

struct node\* deletebeg( struct node\*head ){

struct node\*p;

p=head;

head=head->next;

if(head){

head->prev=NULL;

}

free(p);

return head;

}

struct node\* deleteend(struct node\*head){

struct node\*p;

p=head;

while(p->next!=NULL){

p=p->next;

}

if(p==head){

head==NULL;

}

else{

p->prev->next=NULL;

}

free(p);

return head;

}

struct node\*deletenode( struct node\* head,int val){

if(head==NULL){

cout<<"list is empty";

}

else{

struct node\*p;

p=head;

while(p!=NULL && p->data!=val){

p=p->next;

}

if (temp == NULL){

cout<< "element not found";}

else{

p->prev->next=p->next;

p->next->prev=p->prev;

free(p);}

}

return head;

}

int main(){

struct node\*head;

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

struct node\*second;

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

struct node\*third;

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

struct node\*fourth;

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

head->data=7;

head->next=second;

head->prev=NULL;

second->data=11;

second->next=third;

second->prev=head;

third->data=66;

third->next=fourth;

third->prev=second;

fourth->data=23;

fourth->next=NULL;

fourth->prev=third;

head=deletenode(head,66);

traversal(head);

}

Q2

#include <iostream>

using namespace std;

struct node{

int data;

struct node\*next;

};

void size(struct node\*head){

struct node\*p;

int count=0;

p=head;

while(p!=NULL){

count++;

p=p->next;

}

cout<<"size of list is:"<<count;

}

void display( struct node\*head){

struct node\*p;

cout<<head->data<<" ";

p=head->next;

do{

cout<<p->data<< " ";

p=p->next;

}while(p!=head->next);

}

int main()

{

struct node\*head;

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

struct node\*second;

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

struct node\*third;

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

struct node\*fourth;

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

head->data=7;

head->next=second;

second->data=11;

second->next=third;

third->data=66;

third->next=fourth;

fourth->data=23;

fourth->next=head;

display(head);

return 0;

}

Q3 (a)

#include <iostream>

using namespace std;

struct node{

char data;

struct node\*next;

struct node\*prev;

};

void size(struct node\*head){

struct node\*p;

int count=0;

p=head;

while(p!=NULL){

count++;

p=p->next;

}

cout<<"size of list is:"<<count;

}

int main()

{

struct node\*head;

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

struct node\*second;

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

struct node\*third;

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

struct node\*fourth;

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

head->data=7;

head->next=second;

head->prev=NULL;

second->data=11;

second->next=third;

second->prev=head;

third->data=66;

third->next=fourth;

third->prev=second;

fourth->data=23;

fourth->next=NULL;

fourth->prev=third;

size(head);

return 0;

}

(b)

#include <iostream>

using namespace std;

struct node{

int data;

struct node\*next;

};

void traversal(struct node\*p,struct node\*ptr){

while(ptr->next!=p){

cout<<ptr->data<<" ";

ptr=ptr->next;

}

cout<<ptr->data;

}

void size( struct node\*head){

struct node\*ptr=head;

int i=0;

while(ptr->next!=head){

ptr=ptr->next;

i++;

}

cout<<"size of list:"<<i+1;

}

int main()

{

struct node\*head;

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

struct node\*second;

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

struct node\*third;

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

struct node\*fourth;

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

head->data=7;

head->next=second;

second->data=11;

second->next=third;

third->data=66;

third->next=fourth;

fourth->data=23;

fourth->next=head;

size(head);

Q4

#include <iostream>

using namespace std;

struct node{

char data;

struct node\*next;

struct node\*prev;

};

void palindrome( struct node\*head){

struct node\*p;

struct node\*q;

p=head;

q=head;

while( p->next!=NULL){

p=p->next;

}

int flag=1;

while(q!=p && q->prev!=p){

if(q->data!= p->data){

cout<<"Not a palindrome";

flag=0;

break;

}

q=q->next;

p=p->prev;

}

if(flag){

cout<<"It is a palindrome";

}

}

int main()

{

struct node\*head;

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

cout<<"enter the data of head:";

cin>>head->data;

head->next=NULL;

head->prev=NULL;

struct node\*tail;

tail=head;

int n;

cout<<"\nenter the no of node of linked list:";

cin>>n;

for(int i=1;i<n;i++){

tail->next=(struct node\*)malloc (sizeof(struct node));

tail->next->prev=tail;

tail->next->next=NULL;

cout<<"\nenter the data of node:";

cin>>tail->next->data;

tail=tail->next;

}

palindrome(head);

return 0;

}

return 0;

}

Q5

#include <iostream>

using namespace std;

struct node{

int data;

struct node\*next;

};

void traversal(struct node\*p,struct node\*ptr){

while(ptr->next!=p){

cout<<ptr->data<<" ";

ptr=ptr->next;

}

cout<<ptr->data;

}

}

void size( struct node\*head){

struct node\*ptr=head;

int i=0;

while(ptr->next!=head){

ptr=ptr->next;

i++;

}

cout<<"size of list:"<<i+1;

}

void check(struct node\*head){

struct node\*ptr;

ptr=head->next;

while(ptr!=NULL&& ptr!=head){

ptr=ptr->next;

}

if(ptr==head){

cout<<"list is circular";

}

else

cout<<"list is not circular";

}

int main()

{

struct node\*head;

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

struct node\*second;

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

struct node\*third;

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

struct node\*fourth;

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

head->data=7;

head->next=second;

second->data=11;

second->next=third;

third->data=66;

third->next=fourth;

fourth->data=23;

fourth->next=head;

size(head);

check(head);

return 0;

}