**V Venkateswarlu Gupta**

**1602-19-733-119**

**CSE-B**

**Data Structures Lab**

Week – 6

***Prelab Questions***

***1. Pseudocode to find the mid element of a circular linked list.***

A. void Findmid(){

node \*q;

q:=head;

int c=1;

while(q->next!=head){

q=q->next;

c=c+1;

}

q=q->next;

int i=0;

if(c%2==1){

int k;

k=(c-1)/2;

while(i<k){

q=q->next;

i++;

}

Write(q->data);

}

else{

int k=c/2;

while(i<k-1){

q=q->next;

i++;

}

Write(q->data);

Write(q->next->data);

}

}

**2. Write the pseudocode to reverse the elements of a doubly linked list.**

A. void reverse(){

node \*q;

q:=head;

write("The elements in the reversed order are:\n");

write("Reversed list is :\n");

while(q->next!=NULL){

q=q->next;

}

while(q->prev!=NULL){

printf("%d->",q->data);

q=q->prev;

}

Write(q->data);

}

**3.Psudocode to check if the linkedlist is circular.**

**A.**

Bool isCircular(node \*head){

Node \*t:=head;

While(temp!=Null){

If(temp->next==head)

Return True;

Temp=temp->next;

}

Return false;

}

**4. Advantages of DLL over SLL?**

1.DLL can be traversed in both forward and backward direction.  
2.The delete operation in DLL is more efficient if pointer to the node to be deleted is given.  
3.We can quickly insert a new node before a given node.  
4.In SLL, to delete a node, pointer to the previous node is needed. To get this previous node sometimes the list is traversed. In DLL, we can get the previous node using previous pointer of **this** node.

**5.What are disadvantages of circular linked list.**

A.1. It is not easy to reverse the linked list.  
 2. If proper care is not taken, then the problem of infinite loop can occur.  
 3. If we are at a node and need to go back to the previous node, then we can not do it in single step. Instead we have to complete the entire circle by going through the in between nodes and then we will reach the required node.

**6.Represent a circular doubly linkedlist with an example. Write the psedocode for inserting element at the beginning.**

A. void insertion\_beginning() {

struct node \*q,\*temp;

int item;

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

if temp == NULL)

{

write("\nOVERFLOW");

}

else

{

printf("\nEnter Item value");

scanf("%d",&item);

temp ->data=:item;

if(head==NULL)

{

head =: temp;

temp-> next =: head;

temp -> prev =: head;

}

else

{

q = head;

while(q -> next != head)

{

q =: q-> next;

}

q -> next =: temp;

temp -> prev: = q;

head -> prev: = temp;

temp -> next: = head;

head: = temp;

}

write("\nNode inserted\n");

}

}

***Prelab Programs***

***Circular Linked list***

#include<stdlib.h>

#include<stdio.h>

typedef *struct* node{

*int* data;

*struct* node \*link;

}node;

node \*h;

*void* create(){

*int* n,i;

    node \*q;

    q=h;

    printf("No of nodes : ");

    scanf("%d",&n);

    for(i=0;i<n;i++){

        node \*t=(node \*)malloc(sizeof(node));

        scanf("%d",&t->data);

        t->link=NULL;

        if(q==NULL){

            h=t;

            q=t;

        }else{

            q->link=t;

            q=t;

        }

    }

    q->link=h;

}

*void* insert\_Begin(){

    node \*t=(node \*)malloc(sizeof(node));

    printf("Enter node to insert infront : ");

    scanf("%d",&t->data);

    t->link=h;

    node \*q=h;

    while(q->link!=h){

        q=q->link;

    }

    q->link=t;

    h=t;

}

*void* insert\_InBetween(){

*int* pos,i=-1;

    node \*q=NULL,\*t=(node \*)malloc(sizeof(node));

    q=h;

    printf("Enter position (aray indexing) : ");

    scanf("%d",&pos);

    printf("Enter element to be inserted : ");

    scanf("%d",&t->data);

    while(i<pos-2)

    {

        q=q->link;

        i++;

    }

    t->link=q->link;

    q->link=t;

}

*void* insert\_End(){

    node \*q,\*t=(node \*)malloc(sizeof(node));

    printf("Enter node to insert at the end : ");

    scanf("%d",&t->data);

    t->link=h;

    q=h;

    while(q->link!=h){

        q=q->link;

    }

    q->link=t;

}

*void* delete\_Begin(){

    node \*t,\*q;

    t=h;

    // h=h->link;

    q=h;

    while(q->link!=h){

        q=q->link;

    }

    q->link=h->link;

    printf("deleting at Beginning %d \n",h->data);

    h=h->link;

    free(t);

}

*void* delete\_InBetween(){

*int* pos,i=0;

    node \*q=NULL;

    printf("Enter position to delete (array indexing) : ");

    scanf("%d",&pos);

    q=h;

    while(i<pos-1)

    {

        q=q->link;

        i++;

    }

    node \*t;

    t=q->link;

    q->link=q->link->link;

    printf("deleting In Between %d \n",t->data);

    free(t);

}

*void* delete\_End(){

    node \*q,\*t;

    q=h;

    while(q->link->link!=h){

        q=q->link;

    }

    t=q->link;

    q->link=h;

    printf("deleting at End %d \n",t->data);

    free(t);

}

*void* display(){

    node \*q;

    q=h;

*int* i=0;

    //while(q->link!=h)

    while(i<14)

    {

        printf("%d->",q->data);

        q=q->link;

        i++;

    }

    printf("%d\n",q->data);

}

*void* main(){

*short* *int* choice,repeat;

    h=NULL;

     printf("sInGlLy cIrCuLaR lInKeD lIsT oPeRaTiOnS\n");

    printf(" 1 Create Singlly Linked List\n");

    printf(" 2 Insertion At the Beginning\n");

    printf(" 3 Insertion In Between\n");

    printf(" 4 Insertion At the End\n");

    printf(" 5 Deletion At the Beginning\n");

    printf(" 6 Deletion In Between\n");

    printf(" 7 Deletion At the End\n");

    printf(" 8 Display\n");

    printf(" # Exit\n");

    do

    {

        printf("Enter choice : ");

        scanf("%hi",&choice);

        switch(choice)

        {

            case 1:create();

                   break;

            case 2:insert\_Begin();

                    break;

            case 3:insert\_InBetween();

                   break;

            case 4:insert\_End();

                   break;

            case 5:delete\_Begin();

                   break;

            case 6:delete\_InBetween();

                   break;

            case 7:delete\_End();

                   break;

            case 8:display();

                   break;

            default:exit(1);

        }

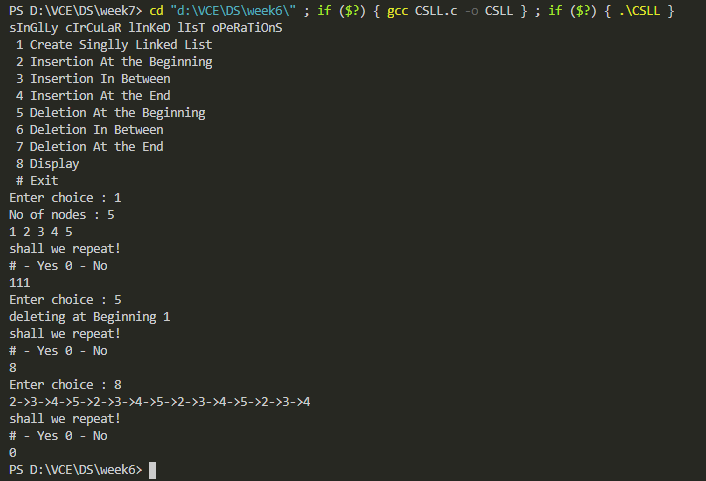
        printf("shall we repeat!\n");

        printf("# - Yes\t0 - No\n");

        scanf("%hi",&repeat);

    }while(repeat);

}



***Doublly Linked list***

#include<stdio.h>

#include<stdlib.h>

typedef *struct* node{

*struct* node \*prev;

*int* data;

*struct* node \*next;

}dll;

dll \*h;

*void* create(){

*int* n,i;

    dll \*q;

    q=h;

    printf("No of nodes : ");

    scanf("%d",&n);

    for(i=0;i<n;i++){

        dll \*t=(dll \*)malloc(sizeof(dll));

        scanf("%d",&t->data);

        if(q==NULL){

            t->prev=NULL;

            h=t;

            q=t;

        }else{

            t->prev=q;

            q->next=t;

            q=t;

        }

    }

    q->next=NULL;

}

*void* insert\_Begin(){

    dll \*t=(dll \*)malloc(sizeof(dll));

    printf("Enter node to insert infront : ");

    scanf("%d",&t->data);

    t->prev=NULL;

    t->next=h;

    h->prev=t;

    h=t;

}

*void* insert\_Inbetween(){

*int* pos,i=-1;

    dll \*q,\*t=(dll \*)malloc(sizeof(dll));

    q=h;

    printf("Enter position (aray indexing)  to insert: ");

    scanf("%d",&pos);

    printf("Enter element to be inserted : ");

    scanf("%d",&t->data);

    while(i<pos-2){

        q=q->next;

        ++i;

    }

    t->prev=q;

    t->next=q->next;

    q->next=t;

    q=t;

    q->next->prev=t;

}

*void* insert\_End(){

    dll \*q,\*t=(dll \*)malloc(sizeof(dll));

    printf("Enter node to insert at the end : ");

    scanf("%d",&t->data);

    while(q->next!=NULL){

        q=q->next;

    }

    t->prev=q;

    q->next=t;

    t->next=NULL;

}

*void* delete\_Begin(){

    dll \*t;

    t=h;

    if(h==NULL){

        printf("No elements to delete\n");

    }else if(h->next==NULL){//Only one element

        free(t);

        h=NULL;

    }else{

        h=h->next;

        h->prev=NULL;

        free(t);

    }

}

*void* delete\_InBetween(){

    dll \*q,\*t;

    q=h;

*int* pos,i=0;

    printf("Enter position to delete (array indexing) : ");

    scanf("%d",&pos);

    while(i<pos-1){

        q=q->next;

        ++i;

    }

    t=q->next;

    q->next=t->next;

    q=q->next;

    q->prev=t->prev;

    free(t);

}

*void* delete\_End(){

    dll \*q,\*t;

    q=h;

    if(h==NULL){

        printf("No elements to delete\n");

    }else if(h->next==NULL){

        t=h;

        free(t);

        h=NULL;

    }else{

        while(q->next->next!=NULL){

            q=q->next;

        }

        t=q->next;

        q->next=NULL;

        free(t);

    }

}

*void* linearSearch(){

*int* x,i=1;

    dll \*q;

    q=h;

    printf("Enter target : ");

    scanf("%d",&x);

    while(q->next!=NULL){

        if(x==q->data){

            printf("Found at %d position\n",i);

            return;

        }

        q=q->next;

        ++i;

    }

    if(x==q->data){

        printf("Found at %d position\n",i);

    }else{

        printf("Not Found\n");

    }

}

*void* bubbleSort()

{

    dll \*i,\*j;

    i=h;

    while(i->next!=NULL)

    {

        j=i->next;

        while(j->next!=NULL)

        {

            if(i->data>j->data)

            {

                i->data=i->data+j->data;

                j->data=i->data-j->data;

                i->data=i->data-j->data;

            }

            j=j->next;

        }

        if(i->data>j->data)

        {

            i->data=i->data+j->data;

            j->data=i->data-j->data;

            i->data=i->data-j->data;

        }

        i=i->next;

    }

}

*void* count(){

    dll \*q;

    q=h;

*int* i=0;

    while(q->next!=NULL){

        i++;

        q=q->next;

    }

    printf("No of nodes : %d",i+1);

}

*void* display(){

    if(h==NULL){

        printf("DLL has no elements\n");

        return;

    }

    dll \*q;

    q=h;

    printf("Forword\n");

    while(q->next!=NULL){

        printf("%d->",q->data);

        q=q->next;

    }

    printf("%d\n",q->data);

    printf("Backwards\n");

    while(q->prev!=NULL){

        printf("%d->",q->data);

        q=q->prev;

    }

    printf("%d\n",q->data);

}

*void* main(){

    h=NULL;

    create();

    display();

    printf("Linear Search\n");

    linearSearch();

    printf("Bubble sort\n");

    bubbleSort();

    display();

    printf("Deletion at begining\n");

    delete\_Begin();

    display();

    delete\_InBetween();

    display();

    insert\_Begin();

    display();

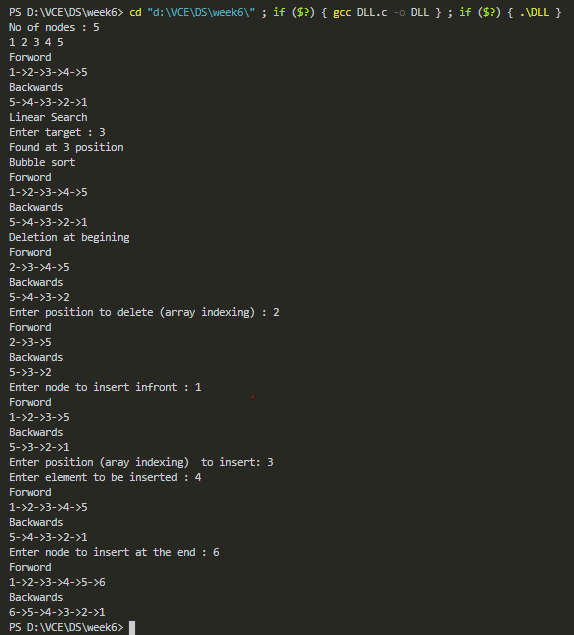
    insert\_Inbetween();

    display();

    insert\_End();

    display();

}



***Employee***

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

*struct* employee

{

*int* id;

*char* name[10];

*float* salary;

*char* dept[10];

};

typedef *struct* node

{

*struct* employee e;

*struct* node \*next;

*struct* node \*prev;

} node;

node \*head = NULL;

node \*head1 = NULL;

node \*head2 = NULL;

*void* createdll()

{

    node \*temp;

    node \*q;

*int* n;

    printf("Number of employees : ");

    scanf("%d", &n);

    printf("Enter Id Name Department Salary %d employees:\n", n);

    for(*int* i=0;i<n;i++){

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

        scanf("%d%s%s%f",&temp->e.id,temp->e.name,temp->e.dept,&temp->e.salary);

        if (head == NULL)

        {

            temp->prev = NULL;

            temp->next = NULL;

            head = temp;

            q = temp;

        }

        else

        {

            q->next = temp;

            temp->prev = q;

            temp->next = NULL;

            q = temp;

        }

    }

}

*void* display()

{

    node \*q;

    q = head;

    while (q != NULL)

    {

        printf("Name:%s\nId:%d\nDept:%s\nSalary:%f\n", q->e.name, q->e.id, q->e.dept, q->e.salary);

        q = q->next;

    }

}

*void* createlist1(node \**q*)

{

    node \*temp, \*r;

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

    if (head1 == NULL)

    {

        temp->prev = NULL;

        temp->next = NULL;

        strcpy(temp->e.name, *q*->e.name);

        temp->e.id = *q*->e.id;

        strcpy(temp->e.dept, *q*->e.dept);

        temp->e.salary = *q*->e.salary;

        r = temp;

        head1 = temp;

    }

    else

    {

        r->next = temp;

        temp->prev = r;

        temp->next = NULL;

        strcpy(temp->e.name, *q*->e.name);

        temp->e.id = *q*->e.id;

        strcpy(temp->e.dept, *q*->e.dept);

        temp->e.salary = *q*->e.salary;

        r = temp;

    }

}

*void* createlist2(node \**q*)

{

    node \*temp, \*r;

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

    if (head2 == NULL)

    {

        temp->prev = NULL;

        temp->next = NULL;

        strcpy(temp->e.name, *q*->e.name);

        temp->e.id = *q*->e.id;

        strcpy(temp->e.dept, *q*->e.dept);

        temp->e.salary = *q*->e.salary;

        r = temp;

        head2 = temp;

    }

    else

    {

        r->next = temp;

        temp->prev = r;

        temp->next = NULL;

        strcpy(temp->e.name, *q*->e.name);

        temp->e.id = *q*->e.id;

        strcpy(temp->e.dept, *q*->e.dept);

        temp->e.salary = *q*->e.salary;

        r = temp;

    }

}

*void* displaylist1()

{

    node \*q;

    q = head1;

    printf("\nEmployees in list 1(CSE/ECE/IT) are:\n");

    if (head1 == NULL)

    {

        printf("\nNo employees exist in list 1\n");

    }

    while (q != NULL)

    {

        printf("Name:%s\nId:%d\nDept:%s\nSalary:%f\n", q->e.name, q->e.id, q->e.dept, q->e.salary);

        q = q->next;

    }

}

*void* displaylist2()

{

    node \*q;

    q = head2;

    printf("\nEmployees in list 2(EEE/MECH/CIVIL) are:\n");

    if (head2 == NULL)

    {

        printf("\nNo employees exist in list 1\n");

    }

    while (q != NULL)

    {

        printf("Name:%s\nId:%d\nDept:%s\nSalary:%f\n", q->e.name, q->e.id, q->e.dept, q->e.salary);

        q = q->next;

    }

}

*void* list()

{

    node \*q = head;

    while (q != NULL)

    {

        if (strcmp(q->e.dept, "CSE") == 0 || strcmp(q->e.dept, "IT") == 0 || strcmp(q->e.dept, "ECE") == 0)

        {

            createlist1(q);

        }

        else if (strcmp(q->e.dept, "EEE") == 0 || strcmp(q->e.dept, "MECH") == 0 || strcmp(q->e.dept, "CIVIL") == 0)

        {

            createlist2(q);

        }

        q = q->next;

    }

}

*void* sortbysalary()

{

    node \*p1, \*p2 = NULL;

    node \*t;

    t = (node \*)malloc(sizeof(node));

*int* swap;

    do

    {

        swap = 0;

        p1 = head;

        while (p1->next != p2)

        {

            if (p1->e.salary > p1->next->e.salary)

            {

                t->e.salary = p1->e.salary;

                strcpy(t->e.name, p1->e.name);

                strcpy(t->e.dept, p1->e.dept);

                t->e.id = p1->e.id;

                p1->e.salary = p1->next->e.salary;

                strcpy(p1->e.name, p1->next->e.name);

                strcpy(p1->e.dept, p1->next->e.dept);

                p1->e.id = p1->next->e.id;

                p1->next->e.salary = t->e.salary;

                strcpy(p1->next->e.name, t->e.name);

                strcpy(p1->next->e.dept, t->e.dept);

                p1->next->e.id = t->e.id;

                swap = 1;

            }

            p1 = p1->next;

        }

        p2 = p1;

    } while (swap);

}

*void* displaynamestartswithS()

{

    node \*q = head;

    printf("\nEmployees whose names start with s:\n");

*short* b = 0;

    while (q != NULL)

    {

        if (strncmp(q->e.name, "s", 1) == 0 || strncmp(q->e.name, "S", 1) == 0)

        {

            printf("Name:%s\nId:%d\nDept:%s\nSalary:%f\n", q->e.name, q->e.id, q->e.dept, q->e.salary);

            b = 1;

        }

        q = q->next;

    }

    if (b == 0)

    {

        printf("\nNo employee exists with initial letter of their name as S\n");

    }

}

*int* main()

{

    createdll();

    printf("\nEmployee details are :\n");

    display();

    list();

    displaylist1();

    displaylist2();

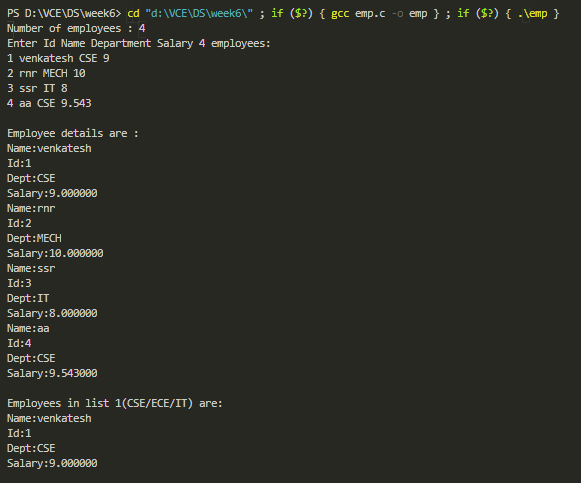
    sortbysalary();

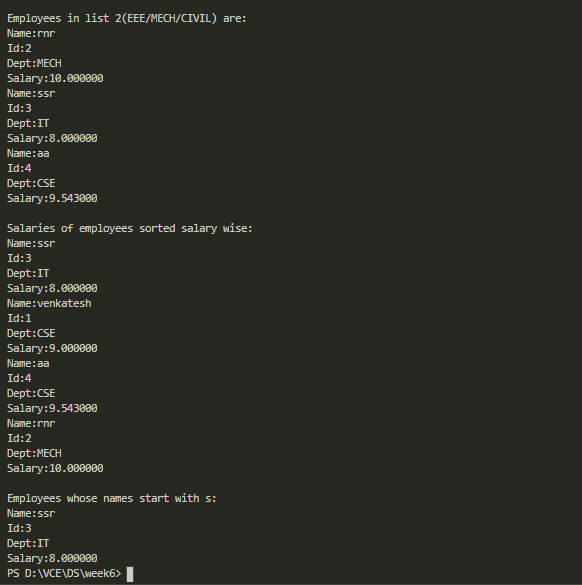
    printf("\nSalaries of employees sorted salary wise:\n");

    display();

    displaynamestartswithS();

}





***Circular Doublly Linked list***

#include<stdio.h>

#include<stdlib.h>

typedef *struct* node{

*struct* node \*prev;

*int* data;

*struct* node \*next;

}cdll;

cdll \*h;

*void* create(){

*int* n,i;

    cdll \*q;

    q=h;

    printf("No of nodes : ");

    scanf("%d",&n);

    for(i=0;i<n;i++){

        cdll \*t=(cdll \*)malloc(sizeof(cdll));

        scanf("%d",&t->data);

        if(q==NULL){

            t->prev=NULL;

            h=t;

            q=t;

        }else{

            t->prev=q;

            q->next=t;

            q=t;

        }

    }

    q->next=h;

    h->prev=q;

}

*void* insert\_Begin(){

    cdll \*q,\*t=(cdll \*)malloc(sizeof(cdll));

    printf("Enter node to insert infront : ");

    scanf("%d",&t->data);

    q=h;

    while(q->next!=h){

        q=q->next;

    }

    q->next=t;

    t->prev=h->prev;

    t->next=h;

    h->prev=t;

    h=t;

}

*void* insert\_Inbetween(){

*int* pos,i=-1;

    cdll \*q,\*t=(cdll \*)malloc(sizeof(cdll));

    q=h;

    printf("Enter position (aray indexing)  to insert: ");

    scanf("%d",&pos);

    printf("Enter element to be inserted : ");

    scanf("%d",&t->data);

    while(i<pos-2){

        q=q->next;

        ++i;

    }

    t->prev=q;

    t->next=q->next;

    q->next=t;

    q=t;

    q->next->prev=t;

}

*void* insert\_End(){

    cdll \*q,\*t=(cdll \*)malloc(sizeof(cdll));

    printf("Enter node to insert at the end : ");

    scanf("%d",&t->data);

    q=h;

    while(q->next!=h){

        q=q->next;

    }

    t->prev=q;

    q->next=t;

    t->next=h;

    h->prev=t;

}

*void* delete\_Begin(){

    cdll \*t,\*q;

    t=h;

    if(h==NULL){

        printf("No elements to delete\n");

    }else if(h->next==h){//Only one element

        free(t);

        h=NULL;

    }else{

        h->next->prev=h->prev;

        q=h;

        while(q->next!=h){

            q=q->next;

        }

        h=h->next;

        q->next=h;

        free(t);

    }

}

*void* delete\_InBetween(){

    cdll \*q,\*t;

    q=h;

*int* pos,i=0;

    printf("Enter position to delete (array indexing) : ");

    scanf("%d",&pos);

    while(i<pos-1){

        q=q->next;

        ++i;

    }

    t=q->next;

    q->next=t->next;

    q=q->next;

    q->prev=t->prev;

    free(t);

}

*void* delete\_End(){

    cdll \*q,\*t;

    q=h;

    if(h==NULL){

        printf("No elements to delete\n");

    }else if(h->next==h){

        t=h;

        free(t);

        h=NULL;

    }else{

        while(q->next->next!=h){

            q=q->next;

        }

        t=q->next;

        q->next=h;

        h->prev=q;

        free(t);

    }

}

*void* linearSearch(){

*int* x,i=1;

    cdll \*q;

    q=h;

    printf("Enter target : ");

    scanf("%d",&x);

    while(q->next!=h){

        if(x==q->data){

            printf("Found at %d position\n",i);

            return;

        }

        q=q->next;

        ++i;

    }

    if(x==q->data){

        printf("Found at %d position\n",i);

    }else{

        printf("Not Found\n");

    }

}

*void* bubbleSort()

{

    cdll \*i,\*j;

    i=h;

    while(i->next!=h)

    {

        j=i->next;

        while(j->next!=h)

        {

            if(i->data>j->data)

            {

                i->data=i->data+j->data;

                j->data=i->data-j->data;

                i->data=i->data-j->data;

            }

            j=j->next;

        }

        if(i->data>j->data)

        {

            i->data=i->data+j->data;

            j->data=i->data-j->data;

            i->data=i->data-j->data;

        }

        i=i->next;

    }

}

*void* display(){

    if(h==NULL){

        printf("CDLL has no elements\n");

        return;

    }

    cdll \*q,\*t;

    q=h;

*int* i=0,j=0;

    printf("Forword\n");

    //while(q->next!=h){

    while(i<12){

        printf("%d->",q->data);

        q=q->next;

        i++;

    }

    t=q;

    printf("%d\n",q->data);

    printf("Backwards\n");

    //while(q->prev!=t){

    while(j<12){

        printf("%d->",q->data);

        q=q->prev;

        j++;

    }

    printf("%d\n",q->data);

}

*void* count(){

    cdll \*q;

    q=h;

*int* i=0;

    if(h==NULL){

        printf("0\n");

        return;

    }

    while(q->next!=h){

        i++;

        q=q->next;

    }

    printf("No of nodes : %d",i+1);

}

*void* main(){

    h=NULL;

    create();

    display();

    // printf("Linear Search\n");

    // linearSearch();

    // printf("Bubble sort\n");

    // bubbleSort();

    // display();

    // insert\_Begin();

    // display();

    // insert\_Inbetween();

    // display();

    // insert\_End();

    // display();

    // printf("Deletion at begining\n");

    // delete\_Begin();

    // display();

    // delete\_InBetween();

    // display();

    // printf("Deletion at End\n");

    // delete\_End();

    // display();

    // count();

}