



**Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM)
A-4, Paschim Vihar, Opp. Paschim Vihar (East) Metro Station, Rohtak Road, New
Delhi, Delhi 110063**

**Data and File Structures - Lab
(MCA-152)
MCA - II Semester**

Enroll no - 01435304419

Semester - II

Section - II

Submitted To - Dr. Sunil Pratap Singh

Submitted By – NIKITA KAPOOR

INDEX

Sno	PROGRAMS	Signature of Teacher
1	Demonstrate the working of a linear linked list.	
2	Implement a doubly linked list with all insertion and deletion operations.	
3	A polynomial is composed of different terms where each of them holds a coefficient and an exponent. Use the list library (list.h) to perform addition of following ploynomials: $4x^4 + 4x^3 - 2x^2 + x$ and $11x^3 + 7x^2 - 4x$.	
4	Write a program that implement stack using static implementation.	
5	Write a program that implement stack using dynamic implementation.	
6	Write a program that implements two stacks (double stack) in a single array using static implementation. Both stacks should grow dynamically in opposite directions, one from lower index to higher index and second from higher to lower index.	
7	Write a program that implements queue using static implementation.	
8	Write a program that implements queue using dynamic implementation.	
9	Implement a circular queue with static implementation.	
10	Implement Doubly Ended queues in a single array.	
11	Develop a binary search tree with dynamic representation.	
12	Develop a heap with static representation and implement a priority queue using heap.	
13	Develop a graph library to implement all specified operations (for integer data elements) using adjacency matrix. Write a program that includes the graph library and calls appropriate graph functions to accept the vertices and edges for the given graph. Display the degree of every vertex, and adjacency matrix of the graph.	

14	Develop a graph library to implement all specified operations (for integer data elements) using adjacency list. Write a program that includes the graph library and calls appropriate graph functions to accept the vertices and edges for the given graph. Display the degree of every vertex of the graph.	
15	Apply Dijkstra's algorithm to determine shortest path from a to f on the below network.	
16	The plan shows a building with five rooms and two doors to the outside. A prize is offered to anyone who can enter the building, pass through every door once only and return to the outside. By modelling the situation, develop a computer-based solution to investigate whether this is possible or not.	
17	Given an array of integers. Implement linear search techniques to search an element in the given array.	
18	Given an array of integers. Implement binary search techniques to search an element in the given array.	
19	Implement Bubble sort algorithm.	
20	Implement Selection sort algorithm.	
21	Implement Insertion sort algorithm.	
22	Implement Shell sort algorithm.	
23	Implement Merge sort algorithm.	
24	Implement Quick sort algorithm.	
25	Implement Heap Sort algorithm.	
26	Implement Hashing.	
27	Write a C program which receives first and last name of 10 students, and then stores the names in a text file "name.txt". After storing the records (names), the program accesses the "name.txt" file to retrieve the students' names and then stores the students' first name in one file "first.txt" and last name in another file "last.txt" in an alphabetical sorted manner.	

Ques 1: Demonstrate the working of a linear linked list.

```

#include<stdio.h>
#include<stdlib.h>

struct node
{
int data;
struct node * ptr;
};

typedef struct node NODE;
NODE *start=NULL;

void insBegin(int val)
{
    NODE *n;
    n=(NODE *)malloc(sizeof(NODE));
    n->data=val;
    if(start==NULL)
    {
        n->ptr=NULL;
        start=n;
    }
    else
    {
        n->ptr=start;
        start=n;
    }
    printf("%d INSERTED",val);
}

void delBegin()
{
    NODE *temp;

    if(start==NULL)
        printf("EMPTY LIST deletion not possible\n");
    else
    {
        temp=start;
        printf("%d DELETED",temp->data);
        start=temp->ptr;
        free(temp);
    }
}

```

```

void insEnd(int val)
{
NODE *temp;
NODE *n;
temp=start;
n=(NODE *)malloc(sizeof(NODE));
if(start==NULL)
{
n->data=val;
n->ptr=NULL;
start=n;
printf("%d INSERTED",val);
return;
}
else
{
while(temp->ptr!=NULL)
{
temp=temp->ptr;
}
temp->ptr=n;
n->data=val;
n->ptr=NULL;
printf("%d INSERTED",val);
}

}

void delEnd()
{
NODE *temp;
NODE *u;
if(start==NULL)
printf("EMPTY LIST deletion not possible\n");
else
{
temp=start;
while(temp->ptr!=NULL)
{
u=temp;
temp=temp->ptr;
}
printf("%d DELETED",temp->data);
free(temp);
u->ptr=NULL;
}
}

```

```

}
void deleteALL()
{
NODE *temp;
temp=start;
while(temp->ptr!=NULL)
{
temp=start;
start=start->ptr;
free(temp);
}
if(start==NULL)
printf("\n LINKED LIST DELETED\n");
}

```

```

void traverse()
{
NODE *temp;
temp=start;
if(start==NULL)
printf("\nLINKED LIST EMPTY\n");
else
{
while(temp!=NULL)
{
printf(" %d ",temp->data);
temp=temp->ptr;
}
}
printf("\n");
}

```

```

void search(int val)
{
NODE *temp;
temp=start;
int c=0;
while(temp!=NULL)
{
if(temp->data==val)
{
c=1;
printf("%d FOUND\n",val);
break;
}
}
}

```

```

    else
        temp=temp->ptr;
    }
    if(c==0)
        printf("%d NOT FOUND\n",val);

    printf("\n");
}

void insPOS(int a,int val)
{
    int i;
    NODE *temp;
    NODE *n;
    int count=0;
    n=(NODE *)malloc(sizeof(NODE));
    temp=start;
    while(temp!=NULL)
    {
        count++;
        temp=temp->ptr;
    }
    if(count==0||count==1)
        insEnd(val);
    if(count>=2)
    {
        if(a>count||a<1)
            printf("Invalid position\n");
    }

    else
    {
        {
            n->data=val;
            temp=start;
            for(i=1;i<a;i++)
            {
                if(i==a-1)
                {
                    n->ptr=temp->ptr;
                    temp->ptr=n;
                }
                temp=temp->ptr;
            }
        }
    }
    printf("%d INSERTED",val);
}

```

```

}
void delPOS(int a)
{
    int count=0;
    int i;
    NODE *temp,*tp;
    if(start==NULL)
        printf("EMPTY LIST deletion not possible\n");
    else
    {
        temp=start;
        while(temp!=NULL)
        {
            count++;
            temp=temp->ptr;
        }
        if(a>count||a<1)
            printf("Invalid position\n");
        if(a==1)
            delBegin();
        else
        {
            temp=start;
            for(i=1;i<=a;i++)
            {
                if(i==a-1)
                {
                    tp=temp->ptr;
                    temp->ptr=tp->ptr;
                    printf("%d DELETED \n",tp->data);
                    if(tp->ptr==NULL)
                        temp->ptr=NULL;
                    free(tp);
                    break;
                }
                temp=temp->ptr;
            }
        }
    }
}

void main()
{

```



```

int ch;
int a,x;
do
{
printf("\nEnter choice...\n");
printf("1.Insert at begin\n");
printf("2.Insert at end\n");
printf("3.Delete at begin\n");
printf("4.Delete at end\n");
printf("5.Delete all\n");
printf("6.Traverse\n");
printf("7.Search\n");
printf("8.Insert at specific pos\n");
printf("9.Delete at specific pos\n");
printf("10.EXIT\n");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("Enter val to enter in linked list:");
scanf("%d",&x);
insBegin(x);
printf("\n");
break;
case 2: printf("Enter val to enter in linked list:");
scanf("%d",&x);
insEnd(x);
printf("\n");
break;
case 3: delBegin();
printf("\n");
break;
case 4:delEnd();
printf("\n");
break;
case 5:deleteALL();
printf("\n");
break;
case 6:printf("\nLinked List:\n");
traverse();
printf("\n");
break;

case 7:printf("Enter val to search in linked list:");
scanf("%d",&x);
search(x);

```

```

        printf("\n");
        break;
    case 8:printf("Enter pos where you want to enter node\n");
        scanf("%d",&a);
        printf("Enter val to enter in linked list:");
        scanf("%d",&x);
        insPOS(a,x);
        break;
    case 9:printf("Enter pos where you want to delete node\n");
        scanf("%d",&a);
        delPOS(a);
    }
}while(ch!=10);
}

```

OUTPUT

```

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Delete all
6.Traverse
7.Search
8.Insert at specific pos
9.Delete at specific pos
10.EXIT
1
Enter val to enter in linked list:1
1 INSERTED

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Delete all
6.Traverse
7.Search
8.Insert at specific pos
9.Delete at specific pos
10.EXIT
2
Enter val to enter in linked list:2
2 INSERTED

```

```

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Delete all
6.Traverse
7.Search
8.Insert at specific pos
9.Delete at specific pos
10.EXIT
8
Enter pos where you want to enter node
2
Enter val to enter in linked list:6
6 INSERTED
Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Delete all
6.Traverse
7.Search
8.Insert at specific pos
9.Delete at specific pos
10.EXIT
6

```

Linked List:

1 6 2

```

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Delete all
6.Traverse
7.Search
8.Insert at specific pos
9.Delete at specific pos
10.EXIT
7
Enter val to search in linked list:1
1 FOUND

```

```

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Delete all
6.Traverse
7.Search
8.Insert at specific pos
9.Delete at specific pos
10.EXIT
3
1 DELETED

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Delete all
6.Traverse
7.Search
8.Insert at specific pos
9.Delete at specific pos
10.EXIT
1
Enter val to enter in linked list:2
2 INSERTED

1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Delete all
6.Traverse
7.Search
8.Insert at specific pos
9.Delete at specific pos
10.EXIT
4
2 DELETED

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Delete all
6.Traverse
7.Search
8.Insert at specific pos
9.Delete at specific pos
10.EXIT
9
Enter pos where you want to delete node
2
6 DELETED

```

```

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Delete all
6.Traverse
7.Search
8.Insert at specific pos
9.Delete at specific pos
10.EXIT
6

Linked List:
 2

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Delete all
6.Traverse
7.Search
8.Insert at specific pos
9.Delete at specific pos
10.EXIT
1
Enter val to enter in linked list:2
2 INSERTED

```

```

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Delete all
6.Traverse
7.Search
8.Insert at specific pos
9.Delete at specific pos
10.EXIT
5

LINKED LIST DELETED

```

Ques2: Implement a doubly linked list with all insertion and deletion operations.

```
#include<stdio.h>
#include<stdlib.h>

struct node
{
int data;
struct node * next;
struct node *prev;
};

typedef struct node NODE;
NODE *head=NULL;
NODE *tail=NULL;

void insBegin(int val)
{
NODE *n;
n=(NODE *)malloc(sizeof(NODE));
n->data=val;
if(head==NULL)
{
n->prev=NULL;
n->next=NULL;
tail=n;
head=n;
}
else
{
n->prev=NULL;
n->next=head;
head=n;
n->next->prev=n;
}
printf("%d INSERTED",val);
}

void delBegin()
{
NODE *temp;

if(head==NULL)
printf("EMPTY LIST deletion not possible\n");
else
{
temp=head;
```

```

    printf("%d DELETED",temp->data);
    head=temp->next;
    free(temp);

}

}
void insEnd(int val)
{
    NODE *n;
    n=(NODE *)malloc(sizeof(NODE));
    n->data=val;
    if(head==NULL)
    {
        n->next=NULL;
        n->prev=NULL;
        head=n;
        tail=n;
        printf("%d INSERTED",val);
        return;
    }
    else
    {
        tail->next=n;
        n->next=NULL;
        n->prev=tail;
        tail=n;
        printf("%d INSERTED",val);
    }

}

void delEnd()
{
    NODE * temp;
    if(tail==NULL)
        printf("EMPTY LIST deletion not possible\n");
    else if(tail->prev==NULL)
    {
        printf("%d DELETED",tail->data);
        tail=tail->next;
        free(tail);
    }
    else
    {
        temp=tail;
        tail=temp->prev;

```

```

    printf("%d DELETED",temp->data);
    free(temp);
    tail->next=NULL;
}
}

void traverse()
{
    NODE *temp;
    temp=head;
    if(head==NULL)
        printf("\nLINKED LIST EMPTY\n");
    else
    {
        while(temp!=NULL)
        {
            printf(" %d ",temp->data);
            temp=temp->next;
        }
    }
    printf("\n");
}

void main()
{

    int ch;
    int a,x;
    do
    {
        printf("\nEnter choice...\n");
        printf("1.Insert at begin\n");
        printf("2.Insert at end\n");
        printf("3.Delete at begin\n");
        printf("4.Delete at end\n");
        printf("5.Traverse\n");
        printf("6.EXIT\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("Enter val to enter in linked list:");
                    scanf("%d",&x);
                    insBegin(x);
                    printf("\n");
                    break;

```



```

    case 2: printf("Enter val to enter in linked list:");
             scanf("%d",&x);
             insEnd(x);
             printf("\n");
             break;
    case 3: delBegin();
             printf("\n");
             break;
    case 4: delEnd();
             printf("\n");
             break;
    case 5: printf("\nLinked List:\n");
             traverse();
             printf("\n");
             break;

        }
    }while(ch!=6);
}

```

OUTPUT

```

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Traverse
6.EXIT
1
Enter val to enter in linked list:1
1 INSERTED

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Traverse
6.EXIT
2
Enter val to enter in linked list:5
5 INSERTED

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Traverse
6.EXIT
5

```

```

Linked List:
 1  5

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Traverse
6.EXIT
4
5 DELETED

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Traverse
6.EXIT
3
1 DELETED

Enter choice...
1.Insert at begin
2.Insert at end
3.Delete at begin
4.Delete at end
5.Traverse
6.EXIT
5

Linked List:

LINKED LIST EMPTY

```

Ques3: A polynomial is composed of different terms where each of them holds a coefficient and an exponent. Use the list library (list.h) to perform addition of following polynomials: $4x^4 + 4x^3 - 2x^2 + x$ and $11x^3 + 7x^2 - 4x$.

```

#include<stdio.h>
#include<stdlib.h>

```

```

struct node
{
int coe;
int exp;
struct node * next;
};

```

```

typedef struct node NODE;

```

```

void create_poly(NODE **start)
{
    int flag;
    int c,e;
    NODE *node;
    node=(NODE *)malloc(sizeof(NODE));
    *start=node;
    do{
        printf("Enter coefficient\n");
        scanf("%d",&c);
        printf("Enter exponent\n");
        scanf("%d",&e);
        node->coe=c;
        node->exp=e;
        node->next=NULL;
        printf("\nWant to add more??\n");
        scanf("%d",&flag);
        if(flag)
        {
            NODE *newnode=(NODE*)malloc(sizeof(NODE));
            node->next=newnode;
            node=newnode;
            node->next=NULL;

        }
    }while(flag);

}

void print_poly(NODE *node)
{
    printf("POLYNOMINAL:\n");
    while(node!=NULL)
    {
        printf(" %d^%d",node->coe,node->exp);
        node=node->next;
        if(node!=NULL)
            printf("+");
    }
    printf("\n");
}

void add_poly(NODE *p,NODE *q,NODE **temp3)
{
    NODE *temp1,*temp2,*node;
    temp1=p;
    temp2=q;
    node=(NODE *)malloc(sizeof(NODE));

```

```

node->next=NULL;
*temp3=node;
while(temp1 && temp2)
{
    if(temp1->exp==temp2->exp)
    {
        node->coe=temp1->coe+temp2->coe;
        node->exp=temp1->exp;

        temp1=temp1->next;
        temp2=temp2->next;
    }
    else if(temp1->exp<temp2->exp)
    {
        node->coe=temp1->coe;
        node->exp=temp1->exp;

        temp1=temp1->next;

    }
    else if(temp1->exp>temp2->exp)
    {
        node->coe=temp2->coe;
        node->exp=temp2->exp;

        temp2=temp2->next;
    }
    if(temp1 && temp2)
    {
        NODE *newnode=(NODE*)malloc(sizeof(NODE));
        node->next=newnode;
        node=newnode;
        node->next=NULL;
    }

}
while(temp1||temp2)
{
    NODE *newnode=(NODE*)malloc(sizeof(NODE));
    node->next=newnode;
    node=newnode;
    node->next=NULL;
    if(temp1)
    {
        node->exp=temp1->exp;

```

```

    node->coe=temp1->coe;
    temp1=temp1->next;
}
if(temp2)
{
    node->exp=temp2->exp;
    node->coe=temp2->coe;
    temp2=temp2->next;
}
}

}
void main()
{

    int ch;
    int x;
    NODE *p,*q,*temp3;
    do
    {
        printf("\nEnter choice...\n");
        printf("1.Create first polynominal: \n");
        printf("2.Print first Polynominal:\n");
        printf("3.Create second polynominal: \n");
        printf("4.Print second Polynominal:\n");
        printf("5.Add Polynominal:\n");
        printf("6.Exit\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:create_poly(&p);
                break;
            case 2:print_poly(p);
                break;
            case 3:create_poly(&q);
                break;
            case 4:print_poly(q);
                break;
            case 5:add_poly(p,q,&temp3);
                print_poly(temp3);
                break;
        }
    }while(ch!=6);
}

```

OUTPUT

```

Enter choice...
1.Create first polynominal:
2.Print first Polynominal:
3.Create second polynominal:
4.Print second Polynominal:
5.Add Polynominal:
6.Exit
1
Enter coeffcient
1
Enter exponent
1

Want to add more??
1
Enter coeffcient
2
Enter exponent
2

Want to add more??
1
Enter coeffcient
3
Enter exponent
3

Want to add more??
0

```

```

Enter choice...
1.Create first polynominal:
2.Print first Polynominal:
3.Create second polynominal:
4.Print second Polynominal:
5.Add Polynominal:
6.Exit
3
Enter coeffcient
1
Enter exponent
1

Want to add more??
1
Enter coeffcient
2
Enter exponent
2

Want to add more??
0

Enter choice...
1.Create first polynominal:
2.Print first Polynominal:
3.Create second polynominal:
4.Print second Polynominal:
5.Add Polynominal:
6.Exit

```

```

6.Exit
2
POLYNOMINAL:
 1^1+ 2^2+ 3^3

Enter choice...
1.Create first polynominal:
2.Print first Polynominal:
3.Create second polynominal:
4.Print second Polynominal:
5.Add Polynominal:
6.Exit
4
POLYNOMINAL:
 1^1+ 2^2

Enter choice...
1.Create first polynominal:
2.Print first Polynominal:
3.Create second polynominal:
4.Print second Polynominal:
5.Add Polynominal:
6.Exit
5
POLYNOMINAL:
 2^1+ 4^2+ 1^1

```

Ques 4: Write a program that implement stack using static implementation.

```

#include<stdio.h>
# define MAX 10
int top=-1;
int stack[MAX];

```

```

int isFull()
{
    if(top==MAX-1)
        return 1;
    else
        return 0;
}

```

```

int isEmpty()
{
    if(top== -1)
        return 1;
    else

```

```
return 0;
printf("\nUnderflow\n");
}

void push(int item)
{
    if(isFull())
        printf("\nOVERFLOW\n");
    else
    {
        top++;
        stack[top]=item;
        printf("%d PUSHED",item);
    }
}

void pop()
{
    if(isEmpty())
        printf("\nUNDERFLOW\n");
    else
    {
        printf(" %d POPPED\n",stack[top]);
        --top;
    }
}

void display()
{
    if(isEmpty())
        printf("\nEMPTY STACK\n");
    else
    {
        printf("STACK :\n");
        for(int i=0;i<=top;i++)
        {
            printf("%d ",stack[i]);
        }
        printf("\n TOP:%d",top);
    }
}

void main()
{
    int ch;
```



```
int x;
do
{
    printf("\nEnter choice...\n");
    printf("1.PUSH\n");
    printf("2.POP\n");
    printf("3.DISPLAY\n");
    printf("4.EXIT\n");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
            printf("Enter val to enter Stack:");
            scanf("%d",&x);
            push(x);
            printf("\n");
            break;
        case 2: pop();
            printf("\n");
            break;
        case 3: display();
            printf("\n");
            break;

    }
}while(ch!=4);
}
```

OUTPUT

```

Enter choice...
1.PUSH
2.POP
3.DISPLAY
4.EXIT
1
Enter val to enter Stack:1
1 PUSHED

Enter choice...
1.PUSH
2.POP
3.DISPLAY
4.EXIT
1
Enter val to enter Stack:2
2 PUSHED

Enter choice...
1.PUSH
2.POP
3.DISPLAY
4.EXIT
3
STACK :
1 2
TOP:1

```

```

Enter choice...
1.PUSH
2.POP
3.DISPLAY
4.EXIT
2
2 POPPED

Enter choice...
1.PUSH
2.POP
3.DISPLAY
4.EXIT
4

```

Ques 5: Write a program that implement stack using dynamic implementation.

```

#include<stdio.h>
#include<stdlib.h>

```

```

struct node
{
int data;
struct node * ptr;
};

```

```

typedef struct node NODE;

```

```

NODE *start=NULL;

void insBeg(int val)
{
    NODE *n;
    n=(NODE *)malloc(sizeof(NODE));
    n->data=val;
    if(start==NULL)
    {
        n->ptr=NULL;
        start=n;
    }
    else
    {
        n->ptr=start;
        start=n;
    }
    printf("%d PUSHED",val);
}

void delBeg()
{
    NODE *temp;

    if(start==NULL)
        printf("EMPTY STACK deletion not possible\n");
    else
    {
        temp=start;
        printf("%d POPPED",temp->data);
        start=temp->ptr;
        free(temp);
    }
}

void Traverse()
{
    NODE *temp;
    temp=start;
    if(start==NULL)
        printf("\nSTACK EMPTY\n");
    else
    {
        while(temp!=NULL)

```

```

{
    printf(" %d ",temp->data);
    temp=temp->ptr;
}
}
printf("\n");
}

```

```

void main()
{

    int ch;
    int a,x;
    do
    {
        printf("\nEnter choice...\n");
        printf("1.PUSH\n");
        printf("2.POP\n");
        printf("3.Display\n");
        printf("4.EXIT\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("Enter val to push in stack:");
                    scanf("%d",&x);
                    insBeg(x);
                    printf("\n");
                    break;
            case 2: delBeg();
                    printf("\n");
                    break;
            case 3: printf(" STACK\n");
                    Traverse();
                    printf("\n");
                    break;

        }
    }while(ch!=4);
}

```

OUTPUT

```

Enter choice...
1.PUSH
2.POP
3.Display
4.EXIT
1
Enter val to push in stack:2
2 PUSHED

Enter choice...
1.PUSH
2.POP
3.Display
4.EXIT
1
Enter val to push in stack:4
4 PUSHED

Enter choice...
1.PUSH
2.POP
3.Display
4.EXIT
2
4 POPPED

```

Ques 6: Write a program that implements two stacks (double stack) in a single array using static implementation. Both stacks should grow dynamically in opposite directions, one from lower index to higher index and second from higher to lower index.

```

#include<stdio.h>
# define MAX 10
int top1=-1;
int top2=MAX;
int stack[MAX];

int isFull()
{
    if(top1==MAX-1)
        return 1;
    if(top2==0)
        return 1;
    if(top1==top2-1||top2==top1+1)
        return 1;
    else
        return 0;
}

```

```

}

int isEmpty1()
{
    if(top1==-1)
        return 1;
    else
        return 0;
}

int isEmpty2()
{
    if(top2==MAX)
        return 1;
    else
        return 0;
}

void push1(int item)
{
    if(isFull())
        printf("\nOVERFLOW\n");
    else
    {
        top1++;
        stack[top1]=item;
        printf("%d PUSHED",item);
    }
}

void pop1()
{
    if(isEmpty1())
        printf("\nUNDERFLOW\n");
    else
    {
        printf(" %d POPPED\n",stack[top1]);
        --top1;
    }
}

void display1()
{
    if(isEmpty1())
        printf("\nEMPTY STACK\n");

```

```

else
{
printf("STACK :\n");
for(int i=0;i<=top1;i++)
{
printf("%d ",stack[i]);
}
printf("\nTOP:%d",top1);
}
}

void push2(int item)
{
if(isFull())
printf("\nOVERFLOW\n");
else
{
--top2;
stack[top2]=item;
printf("%d PUSHED",item);
}
}

void pop2()
{
if(isEmpty2())
printf("\nUNDERFLOW\n");
else
{
printf(" %d POPPED\n",stack[top2]);
top2++;
}
}

void display2()
{
if(isEmpty2())
printf("\nEMPTY STACK\n");
else
{
printf("STACK :\n");
for(int i=MAX-1;i>=top2;i--)
{
printf("%d ",stack[i]);
}
printf("\nTOP:%d",top2);
}
}

```

```

    }
}
void main()
{
int ch;
int a,x;
do
{
printf("\nEnter choice...\n");
printf("1.PUSH IN STACK 1\n");
printf("2.POP IN STACK 1\n");
printf("3.DISPLAY STACK 1\n");
printf("4.PUSH IN STACK 2\n");
printf("5.POP IN STACK 2\n");
printf("6.DISPLAY STACK 2\n");
printf("7.EXIT\n");
scanf("%d",&ch);
switch(ch)
{
case 1: printf("Enter val to enter Stack:");
scanf("%d",&x);
push1(x);
printf("\n");
break;
case 2: pop1();
printf("\n");
break;
case 3: display1();
printf("\n");
break;
case 4: printf("Enter val to enter Stack:");
scanf("%d",&x);
push2(x);
printf("\n");
break;
case 5: pop2();
printf("\n");
break;
case 6: display2();
printf("\n");
break;
}
}while(ch!=7);
}

```

OUTPUT


```
Enter choice...
1.PUSH IN STACK 1
2.POP IN STACK 1
3.DISPLAY STACK 1
4.PUSH IN STACK 2
5.POP IN STACK 2
6.DISPLAY STACK 2
7.EXIT
1
Enter val to enter Stack:1
1 PUSHED

Enter choice...
1.PUSH IN STACK 1
2.POP IN STACK 1
3.DISPLAY STACK 1
4.PUSH IN STACK 2
5.POP IN STACK 2
6.DISPLAY STACK 2
7.EXIT
1
Enter val to enter Stack:2
2 PUSHED

Enter choice...
1.PUSH IN STACK 1
2.POP IN STACK 1
3.DISPLAY STACK 1
4.PUSH IN STACK 2
5.POP IN STACK 2
6.DISPLAY STACK 2
7.EXIT
2
2 POPPED

Enter choice...
1.PUSH IN STACK 1
2.POP IN STACK 1
3.DISPLAY STACK 1
4.PUSH IN STACK 2
5.POP IN STACK 2
6.DISPLAY STACK 2
```

```

5.POP IN STACK 2
6.DISPLAY STACK 2
7.EXIT
3
STACK :
1
TOP:0

Enter choice...
1.PUSH IN STACK 1
2.POP IN STACK 1
3.DISPLAY STACK 1
4.PUSH IN STACK 2
5.POP IN STACK 2
6.DISPLAY STACK 2
7.EXIT
4
Enter val to enter Stack:9
9 PUSHED

Enter choice...
1.PUSH IN STACK 1
2.POP IN STACK 1
3.DISPLAY STACK 1
4.PUSH IN STACK 2
5.POP IN STACK 2
6.DISPLAY STACK 2
7.EXIT
6
STACK :
9
TOP:9

Enter choice...
1.PUSH IN STACK 1
2.POP IN STACK 1
3.DISPLAY STACK 1
4.PUSH IN STACK 2
5.POP IN STACK 2
6.DISPLAY STACK 2
7.EXIT
5
9 POPPED

```

Ques 7: Write a program that implements queue using static implementation.

```

#include<stdio.h>
# define MAX 5
int front=-1,rear=-1;
int q[MAX];

int isfull()
{
    if(rear==MAX-1)
        return 1;
    else
        return 0;
}

int isempty()

```

```

{
    if(front==-1)
        return 1;
    else
        return 0;
}

void enqueue(int val)
{
    if(isfull())
        printf("\nOVERFLOW\n");
    else
    {
        if(front==-1)
            front=0;

        rear++;
        q[rear]=val;
        printf("%d INSERTED",val);
    }
}

void dequeue()
{
    if(isempty())
        printf("\n UNDERFLOW\n");
    else
    {
        printf("%d DEQUEUEUED\n",q[front]);
        front++;

        if(front > rear)
        {
            front=-1;
            rear=-1;
        }
    }
}

void display()
{
    int i;
    if(isempty())
        printf("\nEMPTY QUEUE\n");

```

```

else
{
printf("\nQUEUE\n");
for(i=front;i<=rear;i++)
printf("%d ",q[i]);
printf("\nfront:%d",front);
printf(" rear:%d",rear);
printf("\n");
}
}

```

```

void main()
{
int ch;
int a,x;
do
{
printf("\nEnter choice...\n");
printf("1.ENQUEUE\n");
printf("2.DEQUEUE\n");
printf("3.DISPLAY\n");
printf("4.EXIT\n");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("Enter val to enter Queue:");
scanf("%d",&x);
enqueue(x);
printf("\n");
break;
case 2: dequeue();
printf("\n");
break;
case 3: display();
printf("\n");
break;

}
}while(ch!=4);
}

```

OUTPUT

```
Enter choice...
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
1
Enter val to enter Queue:1
1 INSERTED

Enter choice...
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
1
Enter val to enter Queue:3
3 INSERTED

Enter choice...
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
3

QUEUE
1 3
front:0 rear:1
```

```
Enter choice...
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
2
1 DEQUEUED

Enter choice...
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
3

QUEUE
3
front:1 rear:1
```

Ques 8 : Write a program that implements queue using dynamic implementation.

```
#include<stdio.h>
#include<stdlib.h>

struct node
{
int data;
struct node * ptr;
};

typedef struct node NODE;
NODE *start=NULL;

void insEnd(int val)
{
NODE *temp;
NODE *n;
temp=start;
n=(NODE *)malloc(sizeof(NODE));
if(start==NULL)
{
n->data=val;
n->ptr=NULL;
start=n;
printf("%d inserted",val);
return;
}
else
{
while(temp->ptr!=NULL)
{
temp=temp->ptr;
}
temp->ptr=n;
n->data=val;
n->ptr=NULL;
printf("%d inserted",val);
}

}

void delBeg()
{
NODE *temp;

if(start==NULL)
```

```

printf("EMPTY QUEUE deletion not possible\n");
else
{
    temp=start;
    printf("%d DELETED",temp->data);
    start=temp->ptr;
    free(temp);

}

}

void Traverse()
{
    NODE *temp;
    temp=start;
    if(start==NULL)
        printf("\nQUEUE EMPTY\n");
    else
    {
        while(temp!=NULL)
        {
            printf(" %d ",temp->data);
            temp=temp->ptr;
        }
    }
    printf("\n");
}

void main()
{

    int ch;
    int a,x;
    do
    {
        printf("\nEnter choice...\n");
        printf("1.ENQUEUE\n");
        printf("2.DEQUEUE\n");
        printf("3.Display\n");
        printf("4.EXIT\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("Enter val to enter in queue:");

```

```

        scanf("%d",&x);
        insEnd(x);
        printf("\n");
        break;
    case 2: delBeg();
        printf("\n");
        break;
    case 3: printf(" QUEUE\n");
        Traverse();
        printf("\n");
        break;
}
}while(ch!=4);
}

```

OUTPUT

```

Enter choice...
1.ENQUEUE
2.DEQUEUE
3.Display
4.EXIT
1
Enter val to enter in queue:1
1 inserted

Enter choice...
1.ENQUEUE
2.DEQUEUE
3.Display
4.EXIT
1
Enter val to enter in queue:2
2 inserted

Enter choice...
1.ENQUEUE
2.DEQUEUE
3.Display
4.EXIT
2
1 DELETED

```



```

Enter choice...
1.ENQUEUE
2.DEQUEUE
3.Display
4.EXIT
3
  QUEUE
  2

```

Ques 9 :Implement a circular queue with static implementation.

```

#include<stdio.h>
# define MAX 3
int front=-1,rear=-1;
int q[MAX];

```

```

int isfull()
{
    if(rear==MAX-1&&front==0||front==rear+1)
        return 1;
    else
        return 0;
}

```

```

int isempty()
{
    if(front== -1)
        return 1;
    else
        return 0;
}

```

```

void enqueue(int val)
{
    if(isfull())
        printf("\nOVERFLOW\n");
    else
    {
        if(front== -1)
            front=0;

        rear= (rear+1)%MAX;
        q[rear]=val;
        printf("%d INSERTED\n",val);
    }
}

```

```

    }

}

void dequeue()
{
    if(isempty())
        printf("\n UNDERFLOW\n");
    else
    {
        printf("%d DEQUEUED\n",q[front]);
        if(front==rear)
        {
            front=-1;
            rear=-1;
        }
        else
            front=(front+1)%MAX;
    }
}

void display()
{
    int i;
    if(isempty())
        printf("\n EMPTY QUEUE\n");
    else
    {
        printf("\nQUEUE\n");
        for(i=front;i!=rear;i=(i+1)%MAX)
        {
            printf("%d ",q[i]);
        }
        printf("%d ",q[i]);
        printf("\n front:%d",front);
        printf(" rear:%d",rear);
        printf("\n");
    }
}

void main()
{
    int ch;
    int a,x;
    do
    {
        printf("\nCIRCULAR QUEUE\nEnter choice...\n");

```

```

printf("1.ENQUEUE\n");
printf("2.DEQUEUE\n");
printf("3.DISPLAY\n");
printf("4.EXIT\n");
scanf("%d",&ch);
switch(ch)
{
    case 1:
        printf("Enter val to enter Queue:");
        scanf("%d",&x);
        enqueue(x);
        printf("\n");
        break;
    case 2: dequeue();
        printf("\n");
        break;
    case 3: display();
        printf("\n");
        break;

}
}while(ch!=4);
}

```

OUTPUT

```

CIRCULAR QUEUE
Enter choice...
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
1
Enter val to enter Queue:1
1 INSERTED

CIRCULAR QUEUE
Enter choice...
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
1
Enter val to enter Queue:2
2 INSERTED

CIRCULAR QUEUE
Enter choice...
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
1
Enter val to enter Queue:3

```

```
Enter val to enter Queue:
3 INSERTED
```

```
CIRCULAR QUEUE
Enter choice...
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
3
```

```
QUEUE
1 2 3
front:0 rear:2
```

```
CIRCULAR QUEUE
Enter choice...
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
2
1 DEQUEUED
```

```
CIRCULAR QUEUE
Enter choice...
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
1
Enter val to enter Queue:4
4 INSERTED
```

```
CIRCULAR QUEUE
Enter choice...
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
3
```

```
QUEUE
2 3 4
front:1 rear:0
```

Ques 10 :Implement Doubly Ended queues in a single array.

```

#include<stdio.h>
# define MAX 5
int front=-1,rear=-1;
int q[MAX];

int isfull()
{
    if(rear==MAX-1)
        return 1;
    else
        return 0;
}

int isempty()
{
    if(front== -1)
        return 1;
    else
        return 0;
}

void ins_end(int val)
{
    if(isfull())
        printf("\nOVERFLOW\n");
    else
    {
        if(front== -1)
            front=0;

        rear++;
        q[rear]=val;
        printf("%d INSERTED",val);
    }
}

void ins_beg(int val)
{
    if(front== -1)
    {
        front=0;
        q[++rear]=val;
        printf("%d INSERTED",val);
    }
    else if(front!=0)

```

```

{
    q[--front]=val;
    printf("%d INSERTED",val);
}
else
    printf("INSERTION NOT POSSIBLE\n");

}
void del_front()
{
    if(isempty())
        printf("\n UNDERFLOW\n");
    else
    {
        printf("%d DEQUEUED\n",q[front]);
        front++;

        if(front > rear)
        {
            front=-1;
            rear=-1;
        }
    }

}
void del_end()
{
    if(isempty())
        printf("\n UNDERFLOW\n");
    else
    {
        printf("%d DEQUEUED",q[rear]);
        if(front==rear)
            front=rear=-1;
        else
            rear--;
    }
}
void display()
{
    int i;
    if(isempty())
        printf("\nEMPTY QUEUE\n");
    else
    {
        printf("\nQUEUE\n");

```

```

for(i=front;i<=rear;i++)
printf("%d ",q[i]);
printf("\nfront:%d",front);
printf(" rear:%d",rear);
printf("\n");
}
}

```

```

void main()
{
int ch;
int a,x;
do
{
printf("\nDoubly Ended Queue\n");
printf("Enter choice...\n");
printf("1.INSERT AT END\n");
printf("2.INSERT AT FRONT\n");
printf("3.DELETE AT FRONT\n");
printf("4.DELETE AT END\n");
printf("5.DISPLAY\n");
printf("6.EXIT\n");
scanf("%d",&ch);
switch(ch)
{
case 1: printf("Enter val to enter Queue:");
scanf("%d",&x);
ins_end(x);
printf("\n");
break;
case 2: printf("Enter val to enter Queue:");
scanf("%d",&x);
ins_beg(x);
printf("\n");
break;
case 3: del_front();
printf("\n");
break;
case 4: del_end();
printf("\n");
break;
case 5: display();
printf("\n");
break;

```

```

    }
}while(ch!=6);
}

```

OUTPUT

```

Doubly Ended Queue
Enter choice...
1.INSERT AT END
2.INSERT AT FRONT
3.DELETE AT FRONT
4.DELETE AT END
5.DISPLAY
6.EXIT
2
Enter val to enter Queue:5
5 INSERTED

```

```

Doubly Ended Queue
Enter choice...
1.INSERT AT END
2.INSERT AT FRONT
3.DELETE AT FRONT
4.DELETE AT END
5.DISPLAY
6.EXIT
1
Enter val to enter Queue:9
9 INSERTED

```

```

Doubly Ended Queue
Enter choice...
1.INSERT AT END
2.INSERT AT FRONT
3.DELETE AT FRONT
4.DELETE AT END
5.DISPLAY
6.EXIT
5

QUEUE
5 9
front:0 rear:1

```

```

Doubly Ended Queue
Enter choice...
1.INSERT AT END
2.INSERT AT FRONT
3.DELETE AT FRONT
4.DELETE AT END
5.DISPLAY
6.EXIT
3
5 DEQUEUED

```



```
Doubly Ended Queue
Enter choice...
1.INSERT AT END
2.INSERT AT FRONT
3.DELETE AT FRONT
4.DELETE AT END
5.DISPLAY
6.EXIT
5

QUEUE
9
front:1 rear:1

Doubly Ended Queue
Enter choice...
1.INSERT AT END
2.INSERT AT FRONT
3.DELETE AT FRONT
4.DELETE AT END
5.DISPLAY
6.EXIT
2
Enter val to enter Queue:1
1 INSERTED
```

```
Doubly Ended Queue
Enter choice...
1.INSERT AT END
2.INSERT AT FRONT
3.DELETE AT FRONT
4.DELETE AT END
5.DISPLAY
6.EXIT
5
```

```
QUEUE
1 9
front:0 rear:1
```

```
Doubly Ended Queue
Enter choice...
1.INSERT AT END
2.INSERT AT FRONT
3.DELETE AT FRONT
4.DELETE AT END
5.DISPLAY
6.EXIT
4
9 DEQUEUED
```

```
Doubly Ended Queue
Enter choice...
1.INSERT AT END
2.INSERT AT FRONT
3.DELETE AT FRONT
4.DELETE AT END
5.DISPLAY
6.EXIT
5

QUEUE
1
front:0 rear:0
```

Ques 11: Develop a binary search tree with dynamic representation.

```
#include<stdio.h>
#include<stdlib.h>
#include"stack.h"
struct node
{
```

```

int data;
struct node *left;
struct node *right;
};
typedef struct node NODE;

NODE *create_node(int item)
{
    NODE *node=(NODE *)malloc(sizeof(NODE));
    node->data=item;
    node->left=NULL;
    node->right=NULL;
    return node;
}

NODE* insert(NODE *root,int item)
{
    if(root==NULL)
    {
        return create_node(item);
    }
    else if(root->data == item)
        printf("Already Exist\n");

    else if(root->data > item)
    {
        root->left=insert(root->left,item);
    }
    else if (root->data < item)
        root->right=insert(root->right,item);

    return root;
}

void preorder(NODE *root)
{
    if(root==NULL)
        return;
    printf("%d ",root->data);
    preorder(root->left);
    preorder(root->right);
}

void inorder(NODE *root)
{
    if(root==NULL)

```

```

    return;
    inorder(root->left);
    printf("%d ",root->data);
    inorder(root->right);
}

void postorder(NODE *root)
{
    if(root==NULL)
        return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ",root->data);
}

void inorder_nonr(NODE *root)
{
    NODE *stack[50];
    int top=-1;
    NODE *current=root;
    while(current!=NULL)
    {
        stack[++top]=current;
        current=current->left;
    }

    while(top!=-1)
    {
        current=stack[top--];
        printf(" %d ",current->data);
        current=current->right;
        if(current!=NULL)
        {
            stack[++top]=current;
            current=current->left;
        }
    }
}

void preorder_nonr(NODE * root)
{
    NODE *stack[50];
    int top=-1;
    NODE *p=root;
    if(p==NULL)

```

```

return;

stack[++top]=p;
while(top!=-1)
{
    p=stack[top--];
    printf(" %d ",p->data);
    if(p->right!=NULL)
        stack[++top]=p->right;
    if(p->left!=NULL)
        stack[++top]=p->left;
}

}

void postorder_nonr(NODE * root)
{
    NODE *stack1[50];
    NODE *stack2[50];
    int top1=-1;
    int top2=-1;
    NODE *p=root;
    if(p==NULL)
        return;

    stack1[++top1]=p;
    while(top1!=-1)
    {
        p=stack1[top1--];
        stack2[++top2]=p;
        if(p->left!=NULL)
            stack1[++top1]=p->left;
        if(p->right!=NULL)
            stack1[++top1]=p->right;
    }
    while(top2!=-1)
    {
        p=stack2[top2--];
        printf(" %d ",p->data);
    }
}

void max(NODE *root)
{
    NODE *temp=root;
    while(temp->right!=NULL)
        temp=temp->right;
    printf("Maximum =%d",temp->data);
}

```

```

}
void min(NODE *root)
{
    NODE *temp=root;
    while(temp->left!=NULL)
        temp=temp->left;
    printf("Minimum =%d",temp->data);
}
void height(NODE * root)
{
    int l=0,r=0;
    NODE *temp1=root;
    NODE *temp2=root;
    while(temp1->left !=NULL)
    {
        l++;
        temp1=temp1->left;
    }
    while(temp2->right!=NULL)
    {
        r++;
        temp2=temp2->right;
    }
    if(l>=r)
        printf("Height of Tree: %d",l);
    else
        printf("Height of Tree: %d",r);
}

void m_image(NODE * root)
{
    if(root!=NULL)
    {
        NODE * temp;
        m_image(root->left);
        m_image(root->right);
        temp=root->left;
        root->left=root->right;
        root->right=temp;
    }
}

void main()

```

```

{
    int ch;
    NODE *p;
    NODE *root=NULL;
    int x;
    do
    {
        printf("\n\nEnter choice...\n");
        printf("1.Insert node in Tree\n");
        printf("2.Preorder Traversal (Recursive)\n");
        printf("3.Inorder Traversal (Recursive)\n");
        printf("4.Postorder Traversal (Recursive)\n");
        printf("5.Preorder Traversal (Non Recursive)\n");
        printf("6.Inorder Traversal (Non Recursive)\n");
        printf("7.Postorder Traversal (Non Recursive)\n");
        printf("8.Find Minimum and Maximum Nodes\n");
        printf("9.Height of tree\n");
        printf("10.Image of tree\n");
        printf("11.EXIT\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:printf("Enter Node:");
                    scanf("%d",&x);
                    root=insert(root,x);
                    printf("Inserted %d",x);
                    break;
            case 2:printf("Preorder Traversal\n");
                    preorder(root);
                    printf("\n");
                    break;
            case 3:printf("Inorder Traversal\n");
                    inorder(root);
                    printf("\n");
                    break;
            case 4:printf("Postorder Traversal\n");
                    postorder(root);
                    printf("\n");
                    break;
            case 5:printf("Preorder Traversal\n");
                    preorder_nonr(root);
                    printf("\n");
                    break;
            case 6:printf("Inorder Traversal\n");
                    inorder_nonr(root);
                    printf("\n");

```

```

        break;
    case 7:printf("Postorder Traversal\n");
        postorder_nonr(root);
        printf("\n");
        break;
    case 8:min(root);
        max(root);
        printf("\n");
        break;
    case 9:printf("Height of BST\n");
        height(root);
        printf("\n");
        break;
    case 10:printf("Image of BST\n");
        m_image(root);
        preorder(root);
        printf("\n");
        break;
    }
}while(ch!=11);
}

```

OUTPUT

```

Enter choice...
1.Insert node in Tree
2.Preorder Traversal (Recursive)
3.Inorder Traversal (Recursive)
4.Postorder Traversal (Recursive)
5.Preorder Traversal (Non Recursive)
6.Inorder Traversal (Non Recursive)
7.Postorder Traversal (Non Recursive)
8.Find Minimum and Maximum Nodes
9.Height of tree
10.Image of tree
11.EXIT
1
Enter Node:3
Inserted 3

Enter choice...
1.Insert node in Tree
2.Preorder Traversal (Recursive)
3.Inorder Traversal (Recursive)
4.Postorder Traversal (Recursive)
5.Preorder Traversal (Non Recursive)
6.Inorder Traversal (Non Recursive)
7.Postorder Traversal (Non Recursive)
8.Find Minimum and Maximum Nodes
9.Height of tree
10.Image of tree
11.EXIT
1
Enter Node:1
Inserted 1

```



```

Enter choice...
1.Insert node in Tree
2.Preorder Traversal (Recursive)
3.Inorder Traversal (Recursive)
4.Postorder Traversal (Recursive)
5.Preorder Traversal (Non Recursive)
6.Inorder Traversal (Non Recursive)
7.Postorder Traversal (Non Recursive)
8.Find Minimum and Maximum Nodes
9.Height of tree
10.Image of tree
11.EXIT
1
Enter Node:4
Inserted 4

```

```

Enter choice...
1.Insert node in Tree
2.Preorder Traversal (Recursive)
3.Inorder Traversal (Recursive)
4.Postorder Traversal (Recursive)
5.Preorder Traversal (Non Recursive)
6.Inorder Traversal (Non Recursive)
7.Postorder Traversal (Non Recursive)
8.Find Minimum and Maximum Nodes
9.Height of tree
10.Image of tree
11.EXIT
2
Preorder Traversal
3 1 4

```

```

Enter choice...
1.Insert node in Tree
2.Preorder Traversal (Recursive)
3.Inorder Traversal (Recursive)
4.Postorder Traversal (Recursive)
5.Preorder Traversal (Non Recursive)
6.Inorder Traversal (Non Recursive)
7.Postorder Traversal (Non Recursive)
8.Find Minimum and Maximum Nodes
9.Height of tree
10.Image of tree
11.EXIT
5
Preorder Traversal
3 1 4

```

```

Enter choice...
1.Insert node in Tree
2.Preorder Traversal (Recursive)
3.Inorder Traversal (Recursive)
4.Postorder Traversal (Recursive)
5.Preorder Traversal (Non Recursive)
6.Inorder Traversal (Non Recursive)
7.Postorder Traversal (Non Recursive)
8.Find Minimum and Maximum Nodes
9.Height of tree
10.Image of tree
11.EXIT
3
Inorder Traversal
1 3 4

```

```

Enter choice...
1.Insert node in Tree
2.Preorder Traversal (Recursive)
3.Inorder Traversal (Recursive)
4.Postorder Traversal (Recursive)
5.Preorder Traversal (Non Recursive)
6.Inorder Traversal (Non Recursive)
7.Postorder Traversal (Non Recursive)
8.Find Minimum and Maximum Nodes
9.Height of tree
10.Image of tree
11.EXIT
6
Inorder Traversal
1 3 4

```

```

Enter choice...
1.Insert node in Tree
2.Preorder Traversal (Recursive)
3.Inorder Traversal (Recursive)
4.Postorder Traversal (Recursive)
5.Preorder Traversal (Non Recursive)
6.Inorder Traversal (Non Recursive)
7.Postorder Traversal (Non Recursive)
8.Find Minimum and Maximum Nodes
9.Height of tree
10.Image of tree
11.EXIT
4
Postorder Traversal
1 4 3

```

```

Enter choice...
1.Insert node in Tree
2.Preorder Traversal (Recursive)
3.Inorder Traversal (Recursive)
4.Postorder Traversal (Recursive)
5.Preorder Traversal (Non Recursive)
6.Inorder Traversal (Non Recursive)
7.Postorder Traversal (Non Recursive)
8.Find Minimum and Maximum Nodes
9.Height of tree
10.Image of tree
11.EXIT
7
Postorder Traversal
1 4 3

```

```

Enter choice...
1.Insert node in Tree
2.Preorder Traversal (Recursive)
3.Inorder Traversal (Recursive)
4.Postorder Traversal (Recursive)
5.Preorder Traversal (Non Recursive)
6.Inorder Traversal (Non Recursive)
7.Postorder Traversal (Non Recursive)
8.Find Minimum and Maximum Nodes
9.Height of tree
10.Image of tree
11.EXIT
8
Minimum =1Maximum =4

```

```

Enter choice...
1.Insert node in Tree
2.Preorder Traversal (Recursive)
3.Inorder Traversal (Recursive)
4.Postorder Traversal (Recursive)
5.Preorder Traversal (Non Recursive)
6.Inorder Traversal (Non Recursive)
7.Postorder Traversal (Non Recursive)
8.Find Minimum and Maximum Nodes
9.Height of tree
10.Image of tree
11.EXIT
9
Height of BST
Height of Tree: 1

Enter choice...
1.Insert node in Tree
2.Preorder Traversal (Recursive)
3.Inorder Traversal (Recursive)
4.Postorder Traversal (Recursive)
5.Preorder Traversal (Non Recursive)
6.Inorder Traversal (Non Recursive)
7.Postorder Traversal (Non Recursive)
8.Find Minimum and Maximum Nodes
9.Height of tree
10.Image of tree
11.EXIT
10
Image of BST
3 4 1

```

Ques 12: Develop a heap with static representation and implement a priority queue using heap.

```
#include<stdio.h>
```

```
int n,a[20],j,b[20];
```

```
void heapify(int a[],int n,int i)
```

```
{
    int max=i;
    int l=2*i+1;
    int r=2*i+2;

    if(l<n && a[l]>a[max])
    {
        max=l;
        int temp=a[i];
        a[i]=a[max];
        a[max]=temp;
        heapify(a,n,max);
    }
    if(r<n && a[r]>a[max])
    {
        max=r;
```

```

    int temp=a[i];
    a[i]=a[max];
    a[max]=temp;
    heapify(a,n,max);
}
}
void build(int a[],int n)
{
    int i;
    for(i=n/2-1;i>=0;i--)
        heapify(a,n,i);
}
void print(int a[],int n)
{
    int i;
    for( i=0;i<n;i++)
        printf(" %d ",a[i]);

}

void max_del(int a[])
{
    if(n==0)
    {
        printf("Empty heap\n");
        return;
    }
    int i=a[0];
    printf(" Deleted :%d\n",i);
    a[0]=a[n-1];
    n--;
    heapify(a,n,0);
    print(a,n);
}

void main()
{
    int i;
    int ch;
    printf("Enter no. of nodes you want to add\n");
    scanf("%d",&n);
    printf("Enter nodes\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    printf("\n1.Build Heap \n");
    printf("2.Exit\n");
}

```

```

scanf("%d",&ch);
if(ch==1)
{
    build(a,n);
    printf("Max heap \n");
    print(a,n);
    do
    {
        printf("\n1.Delete\n");
        printf("3.Exit\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:max_del(a);
                break;
        }
    }while(ch!=3);
}
if(ch==2)
return;
}

```

OUTPUT

```

Enter no. of nodes you want to add
5
Enter nodes
4 9 2 11 15

1.Build Heap
2.Exit
1
Max heap
15 11 2 4 9
1.Delete
3.Exit
1
Deleted :15
11 9 2 4
1.Delete
3.Exit
1
Deleted :11
9 4 2
1.Delete
3.Exit
3

```

Ques 13 :Develop a graph library to implement all specified operations (for integer data elements) using adjacency matrix. Write a program that includes the graph library and calls appropriate graph functions to accept the vertices and edges for the given graph. Display the degree of every vertex, and adjacency matrix of the graph.

```
#include<stdio.h>
```

```
int v=6,e,i,j;
```

```
int a[20][20]={ { 1,0,0,0,0,1},{0,0,1,0,0,1},{0,0,0,1,1,1},
                {0,0,1,0,1,0},{0,0,0,1,0,1},{1,1,0,1,1,0}};
```

```
void addVertex()
```

```
{
    v++;
    for(i=1;i<=v;i++)
        a[v][i]=0;
    printf("Inserted vertex %d\n",v);
}
```

```
void addEdge(int s,int d)
```

```
{
    if(s>v && d>v)
    {
        printf("Invalid\n");
        return;
    }

    for(i=0;i<v;i++)
    {
        for(j=0;j<v;j++)
        {
            if(i==(s-1) && j==(d-1))
            {
                a[i][j]=1;
                a[j][i]=1;
            }
        }
    }
    printf("Inserted edge (%d,%d)\n",s,d);
}
```

```
void delVertex(int x)
```

```
{
    if(x>v)
    {
        printf("Invalid\n");
        return;
    }
    x=x-1;
    for(i=0;i<v;i++)
    {
        a[x][i]=0;
        a[i][x]=0;
    }
}
```

```

}

void delEdge(int s,int d)
{
    if(s>v && d>v)
    {
        printf("Invalid\n");
        return;
    }
    for(i=0;i<v;i++)
    {
        for(j=0;j<v;j++)
        {
            if(i==(s-1) && j==(d-1))
            {
                a[i][j]=0;
                a[j][i]=0;
            }
        }
    }
    printf("Deleted edge (%d,%d)\n",s,d);
}

void display()
{
    for(i=0;i<=v;i++)
        printf("%d ",i);

    printf("\n");

    for(i=0;i<v;i++)
    {
        printf("%d ",(i+1));
        for(j=0;j<v;j++)
        {
            printf(" %d ",a[i][j]);
        }
        printf("\n");
    }
}

void tell()
{
    int flag=0;
    for(i=0;i<v;i++)
    {
        for(j=0;j<v;j++)

```

```

{
    if(i==j)
    {
        if(a[i][j]==1)
            flag=1;
    }
    if(a[i][j]==2)
        printf("Parallel Edges (%d,%d)\n",i+1,j+1);
    }
}
if(flag==0)
    printf("Simple Graph\n");
else
    printf("Multi graph\n");

```

```

}

```

```

void degree(int n)
{
    int sum=0;
    n=n-1;
    for(i=0;i<n;i++)
    {
        if(a[i][i]==1)
            sum++;
        sum+=a[n][i];
    }
    printf("%d Degree=%d\n",n,sum);
}

```

```

void main()
{
    int a,b,c;
    do
    {
        printf("1.Enter vertex\n");
        printf("2.Enter edge\n");
        printf("3.Delete vertex\n");
        printf("4.Delete edge\n");
        printf("5.Display\n");
        printf("6.About\n");
        printf("7.Degree\n");
        printf("10.Exit\n");
        scanf("%d",&c);
    }
}

```



```

switch(c)
{
case 1:addVertex();
    break;
case 2:printf("Enter vertices where you want to add Edge\n");
    scanf("%d %d",&a,&b);
    addEdge(a,b);
    break;
case 3:printf("Enter vertex to delete\n");
    scanf("%d",&a);
    delVertex(a);
    break;
case 4:printf("Enter vertices where you want to delete Edge\n");
    scanf("%d %d",&a,&b);
    delEdge(a,b);
    break;
case 5:display();
    break;
case 6:tell();
    break;
case 7:printf("Enter vertex\n");
    scanf("%d",&a);
    degree(a);
    break;
}
}while(c !=10);
}

```

OUTPUT

```

1.Enter vertex
2.Enter edge
3.Delete vertex
4.Delete edge
5.Display
6.About
7.Degree
10.Exit
5
0  1  2  3  4  5  6
1  1  0  0  0  0  1
2  0  0  1  0  0  1
3  0  0  0  1  1  1
4  0  0  1  0  1  0
5  0  0  0  1  0  1
6  1  1  0  1  1  0
1.Enter vertex
2.Enter edge
3.Delete vertex
4.Delete edge
5.Display
6.About
7.Degree
10.Exit
7
Enter vertex
2
2 Degree=2
1.Enter vertex
2.Enter edge
3.Delete vertex
4.Delete edge
5.Display
6.About
7.Degree
10.Exit
1
Inserted vertex 7
1.Enter vertex
2.Enter edge
3.Delete vertex
4.Delete edge
5.Display
6.About

```

```

6.About
7.Degree
10.Exit
2
Enter vertices where you want to add Edge
7 1
Inserted edge (7,1)
1.Enter vertex
2.Enter edge
3.Delete vertex
4.Delete edge
5.Display
6.About
7.Degree
10.Exit
3
Enter vertex to delete
2
1.Enter vertex
2.Enter edge
3.Delete vertex
4.Delete edge
5.Display
6.About
7.Degree
10.Exit
5
0 1 2 3 4 5 6 7
1 1 0 0 0 0 1 1
2 0 0 0 0 0 0 0
3 0 0 0 1 1 1 0
4 0 0 1 0 1 0 0
5 0 0 0 1 0 1 0
6 1 0 0 1 1 0 0
7 1 0 0 0 0 0 0

```

```

1.Enter vertex
2.Enter edge
3.Delete vertex
4.Delete edge
5.Display
6.About
7.Degree
10.Exit
2
Enter vertices where you want to add Edge
2
1
Inserted edge (2,1)
1.Enter vertex
2.Enter edge
3.Delete vertex
4.Delete edge
5.Display
6.About
7.Degree
10.Exit
3
Enter vertex to delete
1
1.Enter vertex
2.Enter edge
3.Delete vertex
4.Delete edge
5.Display
6.About
7.Degree
10.Exit
5
0 1 2 3 4 5 6 7
1 0 0 0 0 0 0 0
2 0 0 1 0 0 1 0
3 0 0 0 1 1 1 0
4 0 0 1 0 1 0 0
5 0 0 0 1 0 1 0
6 0 1 0 1 1 0 0
7 0 0 0 0 0 0 0
1.Enter vertex

```

Ques 14: Develop a graph library to implement all specified operations (for integer data elements) using adjacency list. Write a program that includes the graph library and calls appropriate graph functions to accept the vertices and edges for the given graph. Display the degree of every vertex of the graph.

```
#include <stdio.h>
#include <stdlib.h>

#define N 6

struct Graph {
    struct Node* head[N];
};

struct Node {
    int dest;
    struct Node* next;
};

struct Edge {
    int src, dest;
};

struct Graph* createGraph(struct Edge edges[], int n)
{
    unsigned i;

    struct Graph* graph = (struct Graph*)malloc(sizeof(struct Graph));

    for (i = 0; i < N; i++)
        graph->head[i] = NULL;

    for (i = 0; i < n; i++)
    {
        int src = edges[i].src;
        int dest = edges[i].dest;

        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->dest = dest;

        newNode->next = graph->head[src];

        graph->head[src] = newNode;
    }

    return graph;
}
```

```

}
```

```

void printGraph(struct Graph* graph)
{
    int i;
    for (i = 0; i < N; i++)
    {
        struct Node* ptr = graph->head[i];
        while (ptr != NULL)
        {
            printf("(%d -> %d)\t", i, ptr->dest);
            ptr = ptr->next;
        }

        printf("\n");
    }
}

```

```

void degree(struct Graph* graph)
{
    int i;
    for (i = 0; i < N; i++)
    {
        int count=0;
        struct Node* ptr = graph->head[i];
        while (ptr != NULL)
        {
            count++;
            ptr = ptr->next;
        }
        printf("Degree of %d node is %d",i,count);
        printf("\n");
    }
}

```

```

int main(void)
{
    struct Edge edges[] =
    {
        { 0, 1 }, { 1, 2 }, { 2, 0 }, { 2, 1 },
        { 3, 2 }, { 4, 5 }, { 5, 4 }
    };

    int n = sizeof(edges)/sizeof(edges[0]);

    struct Graph *graph = createGraph(edges, n);
}

```

```

    printGraph(graph);
    degree(graph);
    return 0;
}

```

OUTPUT

```

(0 -> 1)
(1 -> 2)
(2 -> 1)      (2 -> 0)
(3 -> 2)
(4 -> 5)
(5 -> 4)
Degree of 0 node is 1
Degree of 1 node is 1
Degree of 2 node is 2
Degree of 3 node is 1
Degree of 4 node is 1
Degree of 5 node is 1

-----
Process exited after 0.06187 seconds with return value 0
Press any key to continue . . .

```

Ques 15: Apply Dijkstra's algorithm to determine shortest path from a to f on the below network.

```

#include<stdio.h>
#include<conio.h>
#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()
{
    int G[MAX][MAX],i,j,n,u;
    printf("Enter no. of vertices:");
    scanf("%d",&n);
    printf("\nEnter the adjacency matrix:\n");

    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            scanf("%d",&G[i][j]);

    printf("\nEnter the starting node:");
    scanf("%d",&u);
    dijkstra(G,n,u);

    return 0;
}

```

```

}

void dijkstra(int G[MAX][MAX],int n,int startnode)
{
    int cost[MAX][MAX],distance[MAX],pred[MAX];
    int visited[MAX],count,mindistance,nextnode,i,j;

    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            if(G[i][j]==0)
                cost[i][j]=INFINITY;
            else
                cost[i][j]=G[i][j];

    for(i=0;i<n;i++)
    {
        distance[i]=cost[startnode][i];
        pred[i]=startnode;
        visited[i]=0;
    }

    distance[startnode]=0;
    visited[startnode]=1;
    count=1;

    while(count<n-1)
    {
        mindistance=INFINITY;

        for(i=0;i<n;i++)
            if(distance[i]<mindistance&&!visited[i])
            {
                mindistance=distance[i];
                nextnode=i;
            }

        //check if a better path exists through nextnode
        visited[nextnode]=1;
        for(i=0;i<n;i++)
            if(!visited[i])
                if(mindistance+cost[nextnode][i]<distance[i])
                {
                    distance[i]=mindistance+cost[nextnode][i];
                    pred[i]=nextnode;
                }
    }
}

```

```

        count++;
    }

    for(i=0;i<n;i++)
        if(i!=startnode)
        {
            printf("\nDistance of node%d=%d",i,distance[i]);
            printf("\nPath=%d",i);

            j=i;
            do
            {
                j=pred[j];
                printf("<-%d",j);
            }while(j!=startnode);
        }
    }
}

```

OUTPUT

```

Enter no. of vertices:5

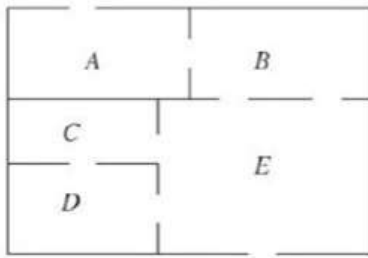
Enter the adjacency matrix:
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0

Enter the starting node:0

Distance of node1=10
Path=1<-0
Distance of node2=50
Path=2<-3<-0
Distance of node3=30
Path=3<-0
Distance of node4=60
Path=4<-2<-3<-0
-----
Process exited after 73.61 seconds with return value 0
Press any key to continue . . .

```

Ques 16: The plan shows a building with five rooms and two doors to the outside. A prize is offered to anyone who can enter the building, pass through every door once only and return to the outside. By modelling the situation, develop a computer-based solution to investigate whether this is possible or not.



Solution: We need to check whether Euler Path Exist or Not

//computer-based solution to investigate whether this is possible or not

```
#include<stdio.h>
```

```
#include<vector>
```

```
#define NODE 5 //as vertices in graph is 5
```

```
using namespace std;
```

```
int graph[NODE][NODE] = { {1, 1, 0, 0, 0},
```

```
    {1, 0, 0, 0, 2},
```

```
    {0, 0, 0, 1, 1},
```

```
    {0, 0, 1, 0, 1},
```

```
    {0, 2, 1, 1, 1}};
```

```
void traverse(int u, bool visited[]) {
```

```
    visited[u] = true; //mark v as visited
```

```
    for(int v = 0; v<NODE; v++) {
```

```
        if(graph[u][v]) {
```

```
            if(!visited[v])
```

```
                traverse(v, visited);
```

```
        }
```

```
    }
```

```
}
```

```
bool isConnected() {
```

```
    bool *vis = new bool[NODE];
```

```
    //for all vertex u as start point, check whether all nodes are visible or not
```

```
    for(int u; u < NODE; u++) {
```

```
        for(int i = 0; i<NODE; i++)
```

```

    vis[i] = false; //initialize as no node is visited
    traverse(u, vis);
    for(int i = 0; i<NODE; i++) {
        if(!vis[i]) //if there is a node, not visited by traversal, graph is not connected
            return false;
    }
}
return true;
}

bool hasEulerPath() {
    int an = 0, bn = 0;
    if(isConnected() == false){ //when graph is not connected
        return false;
    }
    vector<int> inward(NODE, 0), outward(NODE, 0);
    for(int i = 0; i<NODE; i++) {
        int sum = 0;
        for(int j = 0; j<NODE; j++) {
            if(graph[i][j]) {
                inward[j]++; //increase inward edge for destination vertex
                sum++; //how many outward edge
            }
        }
        outward[i] = sum;
    }

    //check the condition for Euler paths
    if(inward == outward) //when number inward edges and outward edges for each node is
    same
        return true; //Euler Circuit, it has Euler path
    for(int i = 0; i<NODE; i++) {
        if(inward[i] != outward[i]) {

```

```

        if((inward[i] + 1 == outward[i])) {
            an++;
        } else if((inward[i] == outward[i] + 1)) {
            bn++;
        }
    }
}

if(an == 1 && bn == 1) { //if there is only an, and bn, then this has euler path
    return true;
}

return false;
}

int main() {
    if(hasEulerPath())
        printf("Euler Path Found.Hence possible to do so!");
    else
        printf("There is no Euler Circuit. Hence not possible to do so!");
}

```

OUTPUT

```

Euler Path Found.Hence possible to do so!
-----
Process exited after 0.09161 seconds with return value 0
Press any key to continue . . . █

```

Ques 17-18: Given an array of integers. Implement linear and binary search techniques to search an element in the given array.

```
#include<stdio.h>
int i;
int linear_search(int a[],int x,int n)
{
    for(i=0;i<n;i++)
    {
        if(a[i]==x)
        {
            printf("Element Found at Position %d",i+1);
            return 1;
        }
    }
    printf("Element Not Found ");
    return 0;
}
void binary_search(int a[],int x,int n)
{
    int low=0;
    int high=n-1;
    int mid;
    while(low<=high)
    {
        mid=(low+high)/2;
        if(a[mid]<x)
            low=mid+1;
        else if(a[mid]>x)
            high=mid-1;
        else
        {
            printf("Element Found at %d",mid+1);
            return;
        }
    }
    printf("Element Not Found\n");
}
int main()
{
    int x;
    int a[]={3,4,8,12,45,67};
    int n=sizeof(a)/sizeof(a[0]);
    for(i=0;i<n;i++)
        printf("%d ",a[i]);

    printf("\nUsing Linear Search\n");
```

```

        printf("\nEnter Element You Want to Search\n");
        scanf("%d",&x);
        linear_search(a,x,n);

        printf("\nEnter Element You Want to Search\n");
        scanf("%d",&x);
        printf("\nUsing Binary Search\n");
        binary_search(a,x,n);

```

```

}

```

OUTPUT

```

3 4 8 12 45 67
Using Linear Search

Enter Element You Want to Search
12
Element Found at Position 4
Enter Element You Want to Search
45

Using Binary Search
Element Found at 5
-----

```

Ques 19-24: Given an array of integers .Sort these integers by implement the following algorithms:

- a) Bubble sort b) Selection sort c) Insertion sort d) Shell sort e) Merge sort f) Quick sort**

```

#include<stdio.h>
int i,j;
void BubbleSort(int a[],int n)
{
    for(i=0;i<=n-1;i++)
    {
        for(j=0;j<=n-i-1;j++)
        {
            if(a[j]>a[j+1])
            {
                int temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
        }
    }
}

```

```

void SelectionSort(int a[],int n)
{
    int loc,temp,min;
    for(i=0;i<n-1;i++)
    {
        min=a[i];
        loc=i;
        for(j=i+1;j<n;j++)
        {
            if(min>a[j])
            {
                min=a[j];
                loc=j;
            }
        }
        temp=a[i];
        a[i]=a[loc];
        a[loc]=temp;
    }
}

void InsertionSort(int a[],int n)
{
    for(i=0;i<n;i++)
    {
        int temp=a[i];
        for(j=i-1;(temp<a[j]&& j>=0);j--)
        {
            a[j+1]=a[j];
        }
        a[j+1]=temp;
    }
}

void ShellSort(int arr[], int n)
{
    for (int gap = n/2; gap > 0; gap /= 2)
    {
        for (int i = gap; i < n; i += 1)
        {
            int temp = arr[i];
            int j;
            for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
                arr[j] = arr[j - gap];
            arr[j] = temp;
        }
    }
}

```

```

void Merge(int a[],int i1,int j1,int i2,int j2)
{
    int tmp[50];
    i=i1;
    j=i2;
    int k=0;
    while(i<=j1&& j<=j2)
    {
        if(a[i]<a[j])
            tmp[k++]=a[i++];
        else
            tmp[k++]=a[j++];
    }
    while(i<=j1)
        tmp[k++]=a[i++];
    while(j<=j2)
        tmp[k++]=a[j++];

    for(i=i1,j=0;i<=j2;i++,j++)
        a[i]=tmp[j];
}

void MergeSort(int a[],int i,int j)
{
    int mid;
    if(i<j)
    {
        mid=(i+j)/2;
        MergeSort(a,i,mid);
        MergeSort(a,mid+1,j);
        Merge(a,i,mid,mid+1,j);
    }
}

int partition (int arr[], int p, int r)
{
    int pivot = arr[r];
    int i = (p - 1);
    for (int j = p; j <= r - 1; j++)
    {
        if (arr[j] < pivot)
        {
            i++;
            int temp=arr[i];
            arr[i]=arr[j];
            arr[j]=temp;
        }
    }
}

```

```

    }
    int temp=arr[i+1];
    arr[i+1]=arr[r];
    arr[r]=temp;

    return (i + 1);
}
void QuickSort(int arr[], int p, int r)
{
    if (p < r)
    {
        int q = partition(arr, p, r);
        QuickSort(arr, p, q - 1);
        QuickSort(arr, q + 1, r);
    }
}
void print(int a[],int n)
{
    for(i=0;i<n;i++)
        printf("%d ",a[i]);
    printf("\n");
}
int main()
{
    int a[10],n,ch;
    do{
        printf("\nEnter Size of Array\n");
        scanf("%d",&n);
        printf("Enter Elements of array\n");
        for(i=0;i<n;i++)
            scanf("%d",&a[i]);

        printf("\nSort Array\n");
        printf("\n1.Bubble Sort");
        printf("\n2.Selection Sort");
        printf("\n3.Insertion Sort");
        printf("\n4.Shell Sort");
        printf("\n5.Merge Sort");
        printf("\n6.Quick Sort");
        printf("\n8.Exit\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:printf("Bubble Sort\n");
                    BubbleSort(a,n);
                    print(a,n);

```



```

        break;
    case 2:printf("Selection Sort\n");
        SelectionSort(a,n);
        print(a,n);
        break;
    case 3:printf("Insertion Sort\n");
        InsertionSort(a,n);
        print(a,n);
        break;
    case 4:printf("Shell Sort\n");
        ShellSort(a,n);
        print(a,n);
        break;
    case 5:printf("Merge Sort\n");
        MergeSort(a,0,n-1);
        print(a,n);
        break;
    case 6:printf("Quick Sort\n");
        QuickSort(a,0,n-1);
        print(a,n);
        break;
    }
}while(ch!=8);

```

}
OUTPUT

```
Enter Size of Array
5
Enter Elements of array
2 3 1 4 5

Sort Array

1.Bubble Sort
2.Selection Sort
3.Insertion Sort
4.Shell Sort
5.Merge Sort
6.Quick Sort
8.Exit
6
Quick Sort
1 2 3 4 5

Enter Size of Array
5
Enter Elements of array
1 2 1 4 5

Sort Array

1.Bubble Sort
2.Selection Sort
3.Insertion Sort
4.Shell Sort
5.Merge Sort
6.Quick Sort
8.Exit
1
Bubble Sort
1 1 2 4 5

Enter Size of Array
4
Enter Elements of array
3 4 6 1
```

```
Sort Array
```

```
1.Bubble Sort
2.Selection Sort
3.Insertion Sort
4.Shell Sort
5.Merge Sort
6.Quick Sort
8.Exit
```

```
2
```

```
Selection Sort
```

```
1 3 4 6
```

```
Enter Size of Array
```

```
4
```

```
Enter Elements of array
```

```
1 5 2 9
```

```
Sort Array
```

```
1.Bubble Sort
2.Selection Sort
3.Insertion Sort
4.Shell Sort
5.Merge Sort
6.Quick Sort
8.Exit
```

```
3
```

```
Insertion Sort
```

```
1 2 5 9
```

```
Enter Size of Array
```

```
5
```

```
Enter Elements of array
```

```
4 5 1 2 8
```

```
Sort Array
```

```
1.Bubble Sort
2.Selection Sort
3.Insertion Sort
4.Shell Sort
5.Merge Sort
6.Quick Sort
8.Exit
```

```
4
```

```
Shell Sort
```

```
1 2 4 5 8
```

```
Enter Size of Array
```

```
5
```

```
Enter Elements of array
```

```
2 3 1 5 4
```

```
Sort Array
```

```
1.Bubble Sort
2.Selection Sort
3.Insertion Sort
4.Shell Sort
5.Merge Sort
6.Quick Sort
8.Exit
```

```
5
```

```
Merge Sort
```

```
1 2 3 4 5
```

Ques 24:HEAP SORT

```

#include<stdio.h>
void heapify(int arr[], int n, int i)
{
    int largest = i;
    int l = 2*i + 1;
    int r = 2*i + 2;

    if (l < n && arr[l] > arr[largest])
        largest = l;

    if (r < n && arr[r] > arr[largest])
        largest = r;
    if (largest != i)
    {
        int temp=arr[i];
        arr[i]=arr[largest];
        arr[largest]=temp;
        heapify(arr, n, largest);
    }
}
void heapSort(int arr[], int n)
{
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);
    for (int i=n-1; i>0; i--)
    {
        int temp=arr[0];
        arr[0]=arr[i];
        arr[i]=temp;
        heapify(arr, i, 0);
    }
}
void printArray(int arr[], int n)
{
    for (int i=0; i<n; ++i)
        printf("%d ",arr[i]);

        printf("\n");
}
int main()
{
    int n,a[20],i;
    printf("\nEnter Size of Array\n");
    scanf("%d",&n);

```

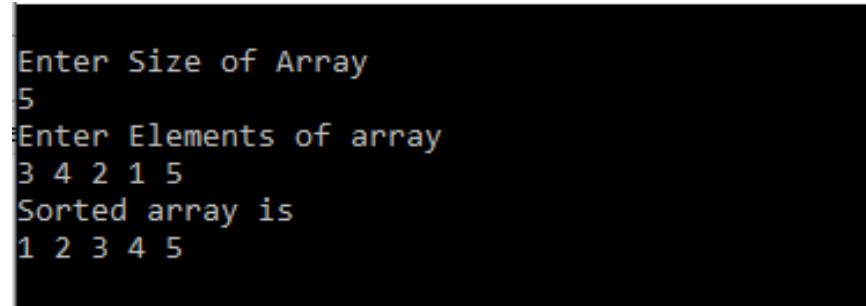
```

printf("Enter Elements of array\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);

heapSort(a, n);
printf("Sorted array is \n");
printArray(a, n);
}

```

OUTPUT



```

Enter Size of Array
5
Enter Elements of array
3 4 2 1 5
Sorted array is
1 2 3 4 5

```

Ques 26 : Hashing

```

#include<stdio.h>
#include<limits.h>

```

```

void insert(int ary[],int hFn, int size){
    int element,pos,n=0;

    printf("Enter key element to insert\n");
    scanf("%d",&element);

    pos = element%hFn;

    while(ary[pos]!= INT_MIN) {
        if(ary[pos]== INT_MAX)
            break;
        pos = (pos+1)%hFn;
        n++;
        if(n==size)
            break;
    }

    if(n==size)
        printf("Hash table was full of elements\nNo Place to insert this element\n\n");
    else
        ary[pos] = element;
}

```

```

void delete(int ary[],int hFn,int size){

```

```

int element,n=0,pos;

printf("Enter element to delete\n");
scanf("%d",&element);

pos = element%hFn;

while(n++ != size){
    if(ary[pos]==INT_MIN){
        printf("Element not found in hash table\n");
        break;
    }
    else if(ary[pos]==element){
        ary[pos]=INT_MAX;
        printf("Element deleted\n\n");
        break;
    }
    else{
        pos = (pos+1) % hFn;
    }
}
if(--n==size)
printf("Element not found in hash table\n");
}

void search(int ary[],int hFn,int size){
    int element,pos,n=0;

    printf("Enter element you want to search\n");
    scanf("%d",&element);

    pos = element%hFn;

    while(n++ != size){
        if(ary[pos]==element){
            printf("Element found at index %d\n",pos);
            break;
        }
        else
        if(ary[pos]==INT_MAX ||ary[pos]!=INT_MIN)
            pos = (pos+1) %hFn;
        }
    if(--n==size) printf("Element not found in hash table\n");
}

```

```

void display(int ary[],int size){
    int i;

    printf("Index\tValue\n");

    for(i=0;i<size;i++)
    {
        if(ary[i]<0)
            printf("%d\t0\n",i);
        else
            printf("%d\t%d\n",i,ary[i]);
    }
}

int main(){
    int size,hFn,i,choice;

    printf("Enter size of hash table\n");
    scanf("%d",&size);

    int ary[size];

    hFn=size;

    for(i=0;i<size;i++)
        ary[i]=INT_MIN;

    do{
        printf("Enter your choice\n");
        printf(" 1-> Insert\n 2-> Delete\n 3-> Display\n 4-> Searching\n 0-> Exit\n");
        scanf("%d",&choice);

        switch(choice){
            case 1:
                insert(ary,hFn,size);
                break;
            case 2:
                delete(ary,hFn,size);
                break;
            case 3:
                display(ary,size);
                break;
            case 4:
                search(ary,hFn,size);
                break;
            default:

```

```

        printf("Enter correct choice\n");
        break;
    }
}while(choice);

return 0;
}

```

```

Enter size of hash table
11
Enter your choice
1-> Insert
2-> Delete
3-> Display
4-> Searching
0-> Exit
1
Enter key element to insert
3
Enter your choice
1-> Insert
2-> Delete
3-> Display
4-> Searching
0-> Exit
1
Enter key element to insert
3
Enter your choice
1-> Insert
2-> Delete
3-> Display
4-> Searching
0-> Exit
3
Index   Value
0       0
1       0
2       0
3       3
4       3
5       0
6       0
7       0
8       0
9       0
10      0
Enter your choice
1-> Insert
2-> Delete
3-> Display
4-> Searching

```



```

Enter your choice
1-> Insert
2-> Delete
3-> Display
4-> Searching
0-> Exit
2
Enter element to delete
3
Element deleted

Enter your choice
1-> Insert
2-> Delete
3-> Display
4-> Searching
0-> Exit
4
Enter element you want to search
3
Element found at index 4
Enter your choice
1-> Insert
2-> Delete
3-> Display
4-> Searching
0-> Exit
0
Enter correct choice

-----
Process exited after 36.97 seconds with return value 0
Press any key to continue . . .

```

Ques 27: Write a C program which receives first and last name of 10 students, and then stores the names in a text file “name.txt”. After storing the records (names), the program accesses the “name.txt” file to retrieve the students’ names and then stores the students’ first name in one file “first.txt” and last name in another file “last.txt” in an alphabetical sorted manner.

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct student
{
    char firstname[20];
    char lastname[20];
}s[10];
int main()
{
    int n;
    int i,j;
    char temp[20],fname[20],lname[20],ch;
    FILE *fout,*fin,*ff,*fl;

```

```

fout=fopen("name.txt","w+");
printf("How Many Students Records You want to enter: ");
scanf("%d",&n);
if(fout==NULL)
{
    fprintf(stderr,"\nError in opening file\n");
    exit(1);
}
/*Writing in name.txt*/
for(i=0;i<n;i++)
{
    printf("\nStudent %d\n",i+1);
    printf("Enter First Name\n");
    scanf("%s",s[i].firstname);
    printf("Enter Last Name\n");
    scanf("%s",s[i].lastname);
}
/*Using Bubble Sort to Sort first name*/
for(i=0;i<=n;i++)
{
    for(j=i+1;j<=n-1;j++)
    {
        if(strcmp(s[i].firstname,s[j].firstname)>0)
        {
            strcpy(temp,s[i].firstname);
            strcpy(s[i].firstname,s[j].firstname);
            strcpy(s[j].firstname,temp);
        }
    }
}
/*Using Bubble Sort to Sort last name*/
for(i=0;i<=n;i++)
{
    for(j=i+1;j<=n-1;j++)
    {
        if(strcmp(s[i].lastname,s[j].lastname)>0)
        {
            strcpy(temp,s[i].lastname);
            strcpy(s[i].lastname,s[j].lastname);
            strcpy(s[j].lastname,temp);
        }
    }
}

fwrite(&s,sizeof(struct student),2,fout);
fin = fopen("name.txt","r");

```

```

        ff=fopen("first.txt","w+");
fl=fopen("last.txt","w+");
/*Reading from name.txt*/
for(i=0;i<n;i++)
{
    fread(&s,sizeof(struct student),1,fin);
    strcpy(fname,s[i].firstname);
    fprintf(ff,"%s",fname);
    strcpy(fname,"\n");
    fprintf(ff,"%s",fname);
}
fclose(ff);
for(i=0;i<n;i++)
{
    fread(&s,sizeof(struct student),1,fin);
    strcpy(lname,s[i].lastname);
    fprintf(fl,"%s",lname);
    strcpy(lname,"\n");
    fprintf(fl,"%s",lname);
}
fclose(fl);
/*Reading First Name from first.txt*/
ff=fopen("first.txt","r");
printf("\nFirst Names\n");
while((ch=fgetc(ff))!=EOF)
    printf("%c",ch);
/*Reading Last Name from last.txt*/
fl=fopen("last.txt","r");
printf("\n\nLast Names\n");
while((ch=fgetc(fl))!=EOF)
    printf("%c",ch);
}

```

OUTPUT

```
How Many Students Records You want to enter: 3

Student 1
Enter First Name
nikita
Enter Last Name
kapoor

Student 2
Enter First Name
kanishka
Enter Last Name
mittal

Student 3
Enter First Name
shalvi
Enter Last Name
gupta

First Names
kanishka
nikita
shalvi

Last Names
gupta
kapoor
mittal
```