**Practical 4**

**1. Write a program to represent a sparse matrix using an array.**

#include <stdio.h>

#define MAX 20

void read\_matrix(int a[10][10], int row, int column);

void print\_sparse(int b[MAX][3]);

void create\_sparse(int a[10][10], int row, int column, int b[MAX][3]);

int main()

{

int a[10][10], b[MAX][3], row, column;

printf("\nEnter the size of matrix (rows, columns): ");

scanf("%d%d", &row, &column);

read\_matrix(a, row, column);

create\_sparse(a, row, column, b);

print\_sparse(b);

return 0;

}

void read\_matrix(int a[10][10], int row, int column)

{

int i, j;

printf("\nEnter elements of matrix\n");

for (i = 0; i < row; i++)

{

for (j = 0; j < column; j++)

{

printf("[%d][%d]: ", i, j);

scanf("%d", &a[i][j]);

}

}

}

void create\_sparse(int a[10][10], int row, int column, int b[MAX][3])

{

int i, j, k;

k = 1;

b[0][0] = row;

b[0][1] = column;

for (i = 0; i < row; i++)

{

for (j = 0; j < column; j++)

{

if (a[i][j] != 0)

{

b[k][0] = i;

b[k][1] = j;

b[k][2] = a[i][j];

k++;

}

}

b[0][2] = k - 1;

}

}

void print\_sparse(int b[MAX][3])

{

int i, column;

column = b[0][2];

printf("\nSparse form - list of 3 triples\n\n");

printf("R\tC\tV\t\n");

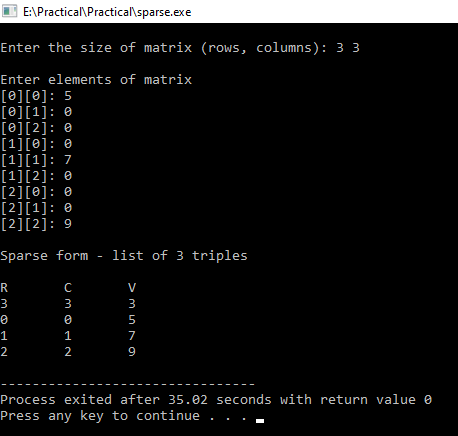
for (i = 0; i <= column; i++)

{

printf("%d\t%d\t%d\n", b[i][0], b[i][1], b[i][2]);

}

}



**2. Write a program to add two sparse matrices using arrays.**

>>#include <stdio.h>

int main ()

{

int arr[20][20];

int m, n;

int count=0;

printf("Enter the order of the matrix : ");

scanf("%d %d", &m, &n);

printf("Enter the values of the matrix: \n");

for (int i=0;i<m;i++)

for (int j=0;j<n;j++)

{

scanf("%d", &arr[i][j]);

if (arr[i][j]!=0)

count++;

}

printf("----------Triplet----------\n");

printf("%d\t%d\t%d\n",m,n,count);

for (int i=0; i<m;i++)

for(int j=0; j<n; j++)

if(arr[i][j]!= 0)

printf("%d\t%d\t%d\n",i,j,arr[i][j]);

int arr2[20][20];

count = 0;

printf("Enter the order of the matrix :");

scanf("%d %d", &m, &n);

printf("Enter the values of the matrix :\n");

for (int i=0;i<m;i++)

for (int j=0;j<n;j++)

{

scanf("%d", &arr2[i][j]);

if (arr2[i][j]!=0)

count++;

}

printf("----------Triplet----------\n");

printf("%d\t%d\t%d\n",m,n,count);

for (int i=0; i<m;i++)

for(int j=0; j<n; j++)

if(arr2[i][j]!= 0)

printf("%d\t%d\t%d\n",i,j,arr2[i][j]);

int arr3[20][20];

printf("----------Additon of Triplet----------\n");

count = 0;

for(int i=0; i<m;i++)

{

for(int j=0; j<n; j++)

arr3[i][j] = arr[i][j]+arr2[i][j];

}

for(int i=0; i<m;i++)

for(int j=0; j<n; j++)

if(arr3[i][j]!= 0)

count++;

printf("%d\t%d\t%d\n",m,n,count);

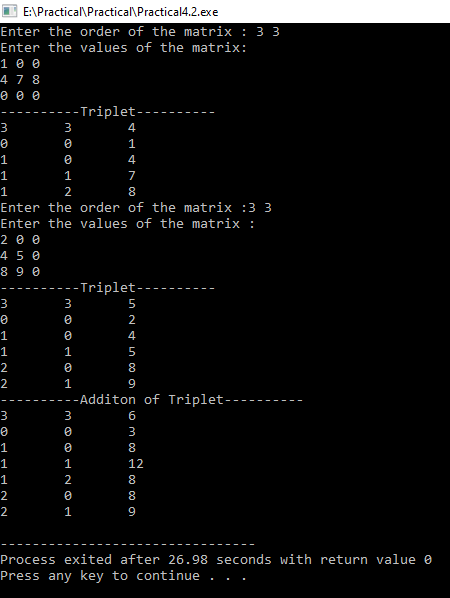
for (int i=0; i<m;i++)

for(int j=0; j<n; j++)

if(arr3[i][j]!= 0)

printf("%d\t%d\t%d\n",i,j,arr3[i][j]);

}



**Practical 8**

**1. Write a program to implement Tower of Hanoi using recursion.**

>>#include<stdio.h>

void toh(int n, char rod1,char rod2,char rod3)

{

if(n==1)

{

printf("\n Move disk 1 from rod %c to rod %c ",rod1,rod2);

return;

}

toh(n-1,rod1,rod3,rod2);

printf("\n Move disk %d from %c to rod %c",n,rod1,rod3);

toh(n-1,rod3,rod2,rod1);

}

int main()

{

int n;

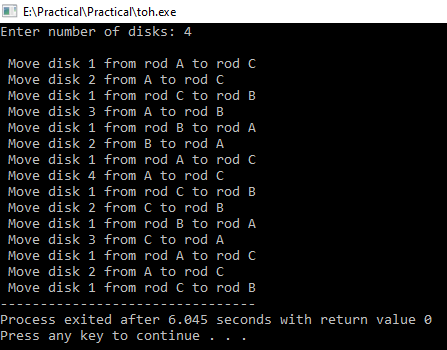
printf("Enter number of disks: ",n);

scanf("%d",&n);

toh(n,'A','B','C');

return 0;

}



**2. Write a program to implement a binary search in array using recursion.**

**>>**#include<stdio.h>

#include<conio.h>

int Search(int A[], int start, int end, int element)

{

if(start>end) return -1;

int mid = (start+end)/2;

if( A[mid] == element ) return mid;

else if( A[mid]>element )

Search(A, start, mid-1, element);

else

Search(A, mid+1, end, element);

}

int main() {

int A[100],n,number,i,j,a;

printf("Enter number of elements: \n",number);

scanf("%d",&number);

for(i=0;i<number;i++)

{

printf("Enter elements: ",A[i]);

scanf("%d",&A[i]);

}

for (i = 0; i < number; ++i)

{

for (j = i + 1; j < number; ++j)

{

if (A[i] > A[j])

{

a = A[i];

A[i] = A[j];

A[j] = a;

}

}

}

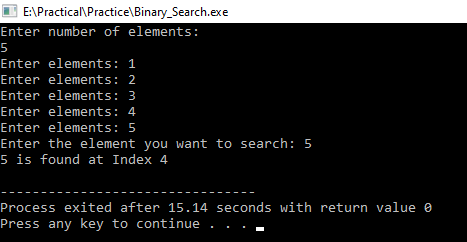
printf("Enter the element you want to search: ",n);

scanf("%d",&n);

printf("%d is found at Index %d \n",n,Search(A,0,number-1,n));

return 0;

}



**3. Write a program to calculate the GCD of two numbers using recursive function.**

**>>**#include <stdio.h>

int GCD(int num1, int num2)

{

if (num1 == 0)

return num2;

if (num2 == 0)

return num1;

if (num1 == num2)

return num1;

if (num1 > num2)

return GCD(num1-num2, num2);

return GCD(num1, num2-num1);

}

int main()

{

int num1,num2;

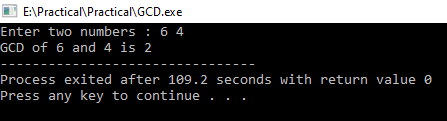
printf("Enter two numbers : ",num1,num2);

scanf("%d %d",&num1,&num2);

printf("GCD of %d and %d is %d",num1,num2,GCD(num1,num2));

return 0;

}



**Practical 9**

**1. Write a program to implement a queue using an array that perform following operations:**

**• Insert**

**• Delete**

**• Display**

**>>**#include<stdio.h>

#define MAX 5

int queue[MAX];

int front=-1;

int rear=-1;

void enqueue(int);

void dequeue();

void display();

int main()

{

int ch,n;

do{

printf("\n Enter your choice :\n 1.Insert\n 2.Delete\n 3.Display\n 4.Exit \n");

scanf("%d",&ch);

switch (ch)

{

case 1:

printf("Enter the element :>>");

scanf("%d",&n);

enqueue(n);

break;

case 2:

dequeue();

break;

case 3:

display();

break;

case 4:

printf("Exiting...");

}

}while(ch != 4);

return 0;

}

void enqueue(int x)

{

if(rear== N-1)

{

printf("Overflow");

}

else if (front==-1 && rear==-1)

{

front=rear=0;

queue[rear]=x;

}

else

{

rear++;

queue[rear]=x;

}

}

void dequeue()

{

if(front==-1 && rear==-1)

{

printf("Underflow");

}

else if(front==rear)

{

front=rear=-1;

}

else

{

printf("dequeue element %d",queue[front]);

front++;

}

}

void display()

{

int i;

if(front==-1 && rear==-1)

{

printf("Queue is Empty");

}

else

{

printf("Elemets are : ");

for(i=front;i<rear+1;i++)

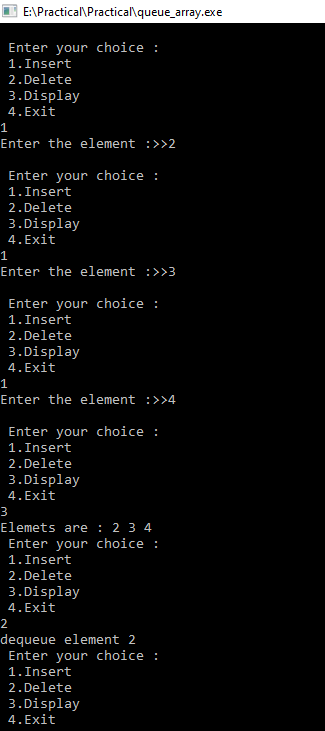
{

printf("%d ",queue[i]);

}

}

}



**2. Write a program to implement a circular queue using an array.**

**>>**#include<stdio.h>

#include<stdlib.h>

#define MAX 5

void insert();

void delet();

void display();

int front = -1, rear = -1;

int queue[MAX];

int main ()

{

int choice;

while(choice != 4)

{

printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");

printf("\nEnter your choice ?");

scanf("%d",&choice);

switch(choice)

{

case 1:

insert();

break;

case 2:

delet();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

printf("\nEnter valid choice??\n");

}

}

return 0;

}

void insert()

{

int item;

printf("\nEnter the element\n");

scanf("%d",&item);

if((rear+1)%MAX == front)

{

printf("\nOVERFLOW");

return;

}

else if(front == -1 && rear == -1)

{

front = 0;

rear = 0;

}

else if(rear == MAX -1 && front != 0)

{

rear = 0;

}

else

{

rear = (rear+1)%MAX;

}

queue[rear] = item;

printf("\nValue inserted ");

}

void delet()

{

int item;

if(front == -1 & rear == -1)

{

printf("\nUNDERFLOW\n");

return;

}

else if(front == rear)

{

front = -1;

rear = -1;

}

else if(front == MAX -1)

{

front = 0;

}

else

front = front + 1;

}

void display()

{

int i;

if(front == -1)

printf("\nCircular Queue is Empty!!!\n");

else

{

i = front;

printf("\nCircular Queue Elements are : \n");

if(front <= rear){

while(i <= rear)

printf("%d\n",queue[i++]);

}

else{

while(i <= MAX - 1)

printf("%d\n", queue[i++]);

i = 0;

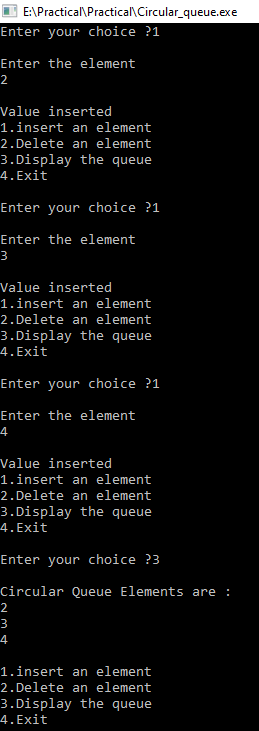
while(i <= rear)

printf("%d\n",queue[i++]);

}

}

}



**3. Write a program to implement a doubly queue using an array.**

#include"stdio.h"

#include"conio.h"

#define MAX 10

int queue[MAX];

int front = -1,rear = -1;

void inputres();

void outputres();

void insertfront(int n);

void insertrear(int n);

void deletefront();

void deleterear();

void display();

int main()

{

int n,i;

do

{

printf("\n-------------------Doubly Queue-------------------\n");

printf("1. Input restricted queue\n");

printf("2. Output restricted queue\n");

printf("3. Exit\n");

printf("Enter the option :");

scanf("%d",&i);

switch(i)

{

case 1:

inputres();

break;

case 2:

outputres();

break;

case 3:

printf("-------------------Thank you-------------------");

break;

default:

printf("Invalid Operation\n");

}

}while(i!=4);

return 0;

}

void insertrear(int n)

{

if((front == 0 && rear == MAX-1) || (front = rear+1))

{

printf("Queue if Full\n");

}

else if(front == -1)

{

front = rear = 0;

queue[rear] = n;

}

else if(front!=0 && rear==MAX-1)

{

rear = 0;

queue[rear] = n;

}

else

{

rear++;

queue[rear] = n;

}

}

void insertfront(int n)

{

if((front == 0 && rear == MAX-1) || (front = rear+1))

{

printf("Queue if Full\n");

}

else if(front == -1)

{

front = rear = 0;

queue[front] = n;

}

else if(front==0 && rear!=MAX-1)

{

front = 0;

queue[front] = n;

}

else

{

front--;

queue[front] = n;

}

}

void deletefront()

{

if(front == -1)

{

printf("Queue is Empty\n");

}

else if(front == rear)

{

front = -1;

rear = -1;

}

else if(front == MAX-1)

{

front =0;

}

else

{

front++;

}

}

void deleterear()

{

if(front == -1)

{

printf("Queue is Empty\n");

}

else if(front == rear)

{

front = -1;

rear = -1;

}

else if(rear == 0)

{

rear = MAX-1;

}

else

{

rear--;

}

}

void display()

{

if(front == -1)

{

printf("Queue is Empty\n");

}

else if(front>rear)

{

for(int i=front;i<MAX-1;i++)

printf("%d \t",queue[i]);

for(int i=0;i<=rear;i++)

printf("%d",queue[i]);

}

else

{

for(int i=front;i<=rear;i++)

printf("%d\t",queue[i]);

}

}

void inputres()

{

int opt, value;

do

{

printf("\n-------------------Input Restricted-------------------\n");

printf("1. Insert element from rear\n");

printf("2. Delete element from front\n");

printf("3. Delete element form rear\n");

printf("4. EXIT\n");

printf("Enter the Option:");

scanf("%d",&opt);

switch(opt)

{

case 1:

printf("Enter the value");

scanf("%d",&value);

insertrear(value);

break;

case 2:

deletefront();

break;

case 3:

deleterear();

break;

case 4:

printf("-------------------Thank you-------------------");

break;

}

}while(opt!=4);

}

void outputres()

{

int opt, value;

do

{

printf("\n-------------------Output Restricted-------------------\n");

printf("1. Insert element from front\n");

printf("2. Insert element from rear\n");

printf("3. Delete element form front\n");

printf("4. EXIT\n");

printf("Enter the Option:");

scanf("%d",&opt);

switch(opt)

{

case 1:

printf("Enter the value");

scanf("%d",&value);

insertfront(value);

break;

case 2:

printf("Enter the value");

scanf("%d",&value);

insertrear(value);

break;

case 3:

deletefront();

break;

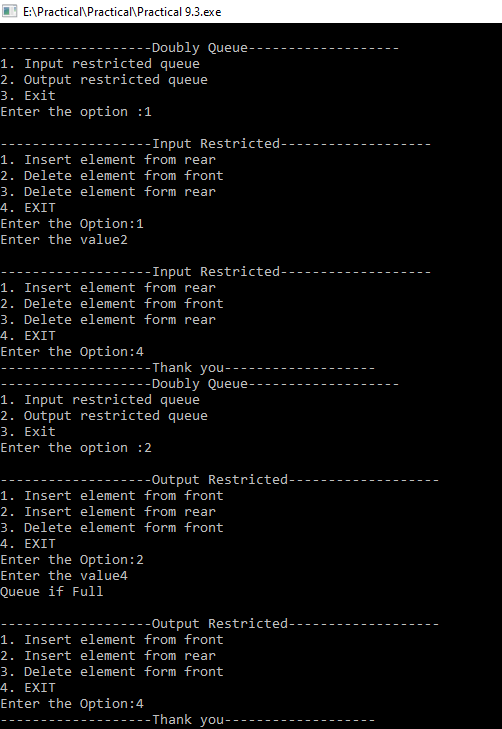
case 4:

printf("-------------------Thank you-------------------");

break;

}

}while(opt!=4);

}

**4. Write a program to implement a priority queue using an array.**

>>#include <stdio.h>

#include <stdlib.h>

#define MAX 5

void insert\_by\_priority(int);

void delete\_by\_priority(int);

void create();

void check(int);

void display\_pqueue();

int pri\_que[MAX];

int front, rear;

int main()

{

int n, ch;

printf("\n1 - Insert an element into queue");

printf("\n2 - Delete an element from queue");

printf("\n3 - Display queue elements");

printf("\n4 - Exit");

create();

while (1)

{

printf("\nEnter your choice : ");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("\nEnter value to be inserted : ");

scanf("%d",&n);

insert\_by\_priority(n);

break;

case 2:

printf("\nEnter value to delete : ");

scanf("%d",&n);

delete\_by\_priority(n);

break;

case 3:

display\_pqueue();

break;

case 4:

exit(0);

default:

printf("\nChoice is incorrect, Enter a correct choice");

}

}

}

void create()

{

front = rear = -1;

}

void insert\_by\_priority(int data)

{

if (rear >= MAX - 1)

{

printf("\nQueue overflow no more elements can be inserted");

return;

}

if ((front == -1) && (rear == -1))

{

front++;

rear++;

pri\_que[rear] = data;

return;

}

else

check(data);

rear++;

}

void check(int data)

{

int i,j;

for (i = 0; i <= rear; i++)

{

if (data >= pri\_que[i])

{

for (j = rear + 1; j > i; j--)

{

pri\_que[j] = pri\_que[j - 1];

}

pri\_que[i] = data;

return;

}

}

pri\_que[i] = data;

}

void delete\_by\_priority(int data)

{

int i;

if ((front==-1) && (rear==-1))

{

printf("\nQueue is empty no elements to delete");

return;

}

for (i = 0; i <= rear; i++)

{

if (data == pri\_que[i])

{

for (; i < rear; i++)

{

pri\_que[i] = pri\_que[i + 1];

}

pri\_que[i] = -99;

rear--;

if (rear == -1)

front = -1;

return;

}

}

printf("\n%d not found in queue to delete", data);

}

void display\_pqueue()

{

if ((front == -1) && (rear == -1))

{

printf("\nQueue is empty");

return;

}

for (; front <= rear; front++)

{

printf(" %d ", pri\_que[front]);

}

front = 0;

}

