

**Session**: 2020-21

**Name:** Sumit Porwal

**Enrollment No:** EN18CS301275

**Class:** CS-E

**Semester:** 6th Sem

**Subject:** Design and Analysis of Algorithms

**Experiment File**

|  |  |  |
| --- | --- | --- |
| **S.no.** | **Experimets** | **Page no.** |
| 1 | Write a program to implement Bubble Sort. | 3-4 |
| 2 | Write a program to implement Insertion Sort. | 5-6 |
| 3 | Write a program to implement Selection Sort. | 7-8 |
| 4 | Write a program to implement Linear Search. | 9-10 |
| 5 | Write a program to implement Binary Search. | 11-12 |
| 6 | Write a program to implement Quick Sort. | 13-15 |
| 7 | Write a program to implement Merge Sort. | 16-18 |
| 8. | Write a program to implement Heap Sort. | 19-21 |
| 9. | Write a program to implement radix sort. | 22-23 |
| 10. | Write a program to implement Binary Search Recursive. | 24-6 |
| 11. | Write a program to implement Strassen’s matrix multiplication. | 27-29 |
| 12. | Write a program to implement optimal merge pattern. | 30-32 |
| 13. | program to implement fractional knapsack problem in c. | 33-35 |
| 14. | Program to implement dijkstras algorithm in c. | 36-39 |
| 15. | Program to implement job sequencing with deadline in c. | 40-42 |
| 16. | Program to implement Kruskal’s algorithm in c. | 43-45 |
| 17. | Program to implement Prim’s algorithm in c. | 46-48 |

**Experiment-I**

**Aim:** Write a program to implement bubble sort.

**Program:**

#include <stdio.h>

int main()

{

int a[50],n,i,j,temp;

printf("Enter the size of array: ");

scanf("%d",&n);

printf("Enter the array elements: ");

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

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

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

{

for(j=0;j<(n-i);++j)

if(a[j]>a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

printf("Array after bubble sort:");

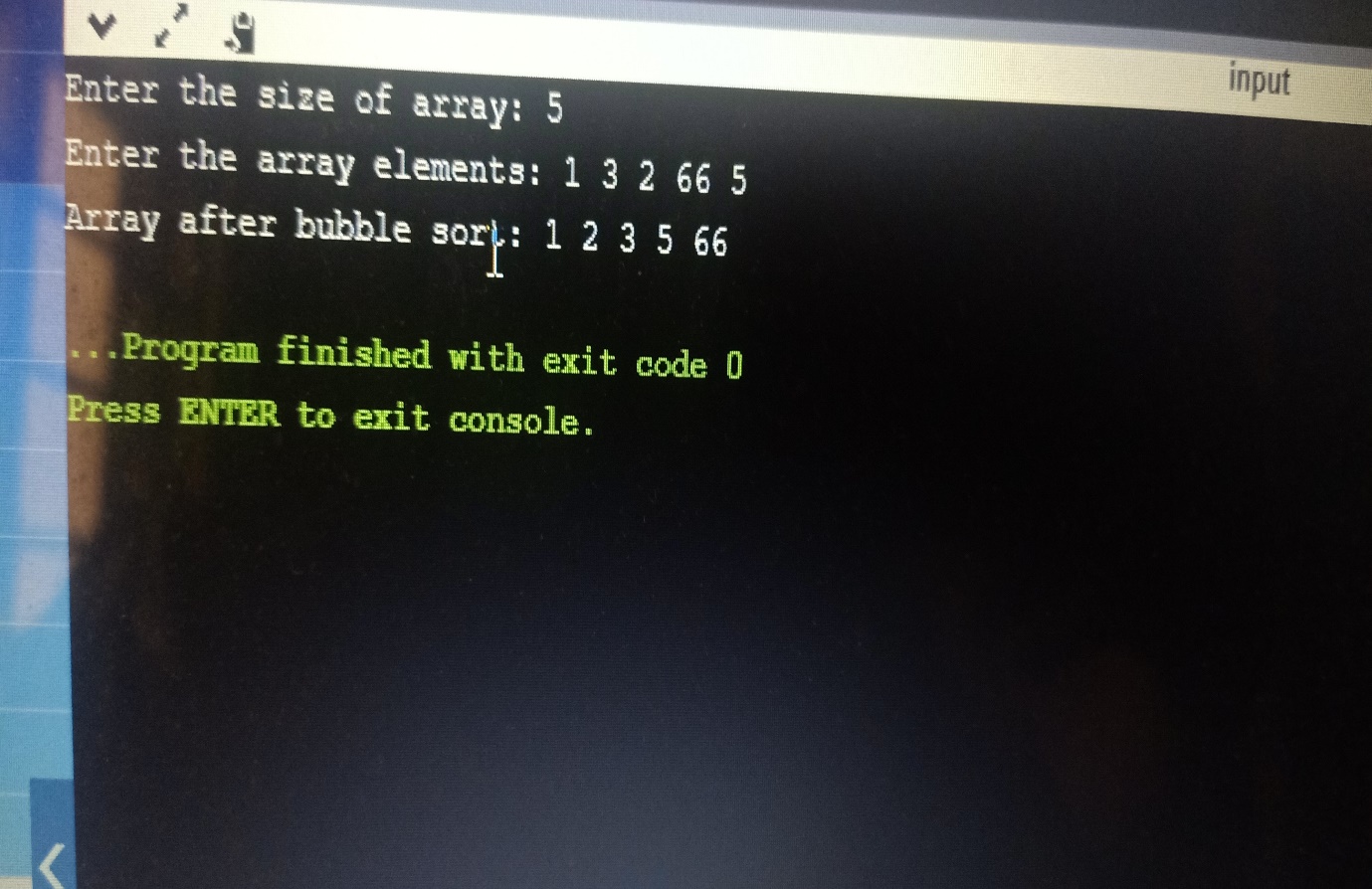
for(i=0;i<n;++i)

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

return 0;

}

**Output:**



**Experiment-II**

**Aim:** Write a program to implement insertion sort.

**Program:**

#include <stdio.h>

int main()

{

int i,j,n,temp,a[30];

printf("Enter the number of elements:");

scanf("%d",&n);

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

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

{

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

}

for(i=1;i<=n-1;i++)

{

temp=a[i];

j=i-1;

while((temp<a[j])&&(j>=0))

{

a[j+1]=a[j];

j=j-1;

}

a[j+1]=temp;

}

printf("\nSorted list after insertion sort is as follows\n");

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

{

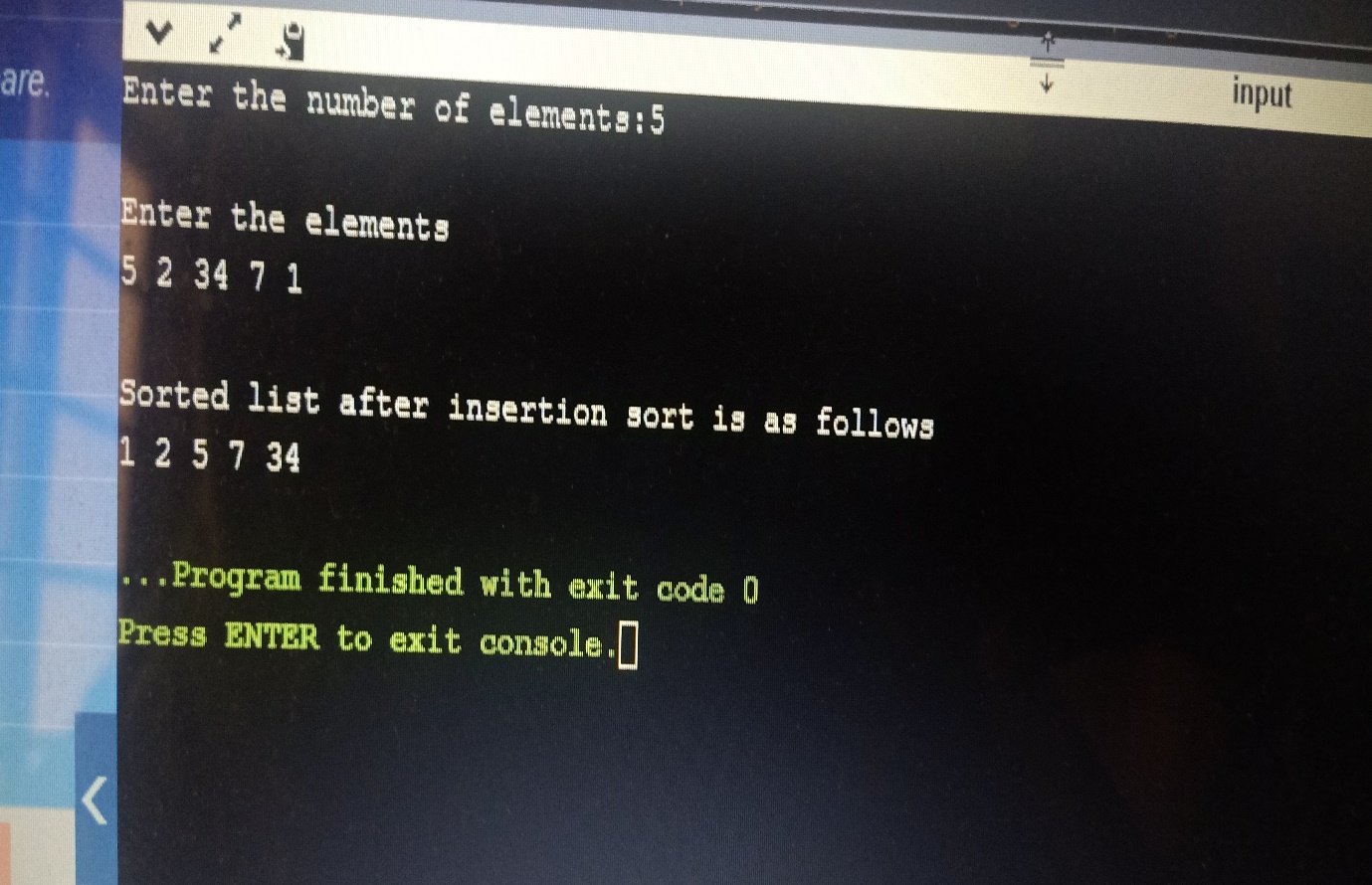
printf("%d ",a[i]);

}

return 0;

}

**Output:**

****

**Experiment-III**

**Aim:** Write a program to implement selection sort.

**Program:**

#include <stdio.h>

int main()

{

int i,j,n,loc,temp,min,a[30];

printf("Enter the number of elements:");

scanf("%d",&n);

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

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

{

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

}

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;

}

printf("\nSorted list after selection sort is as follows\n");

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

{

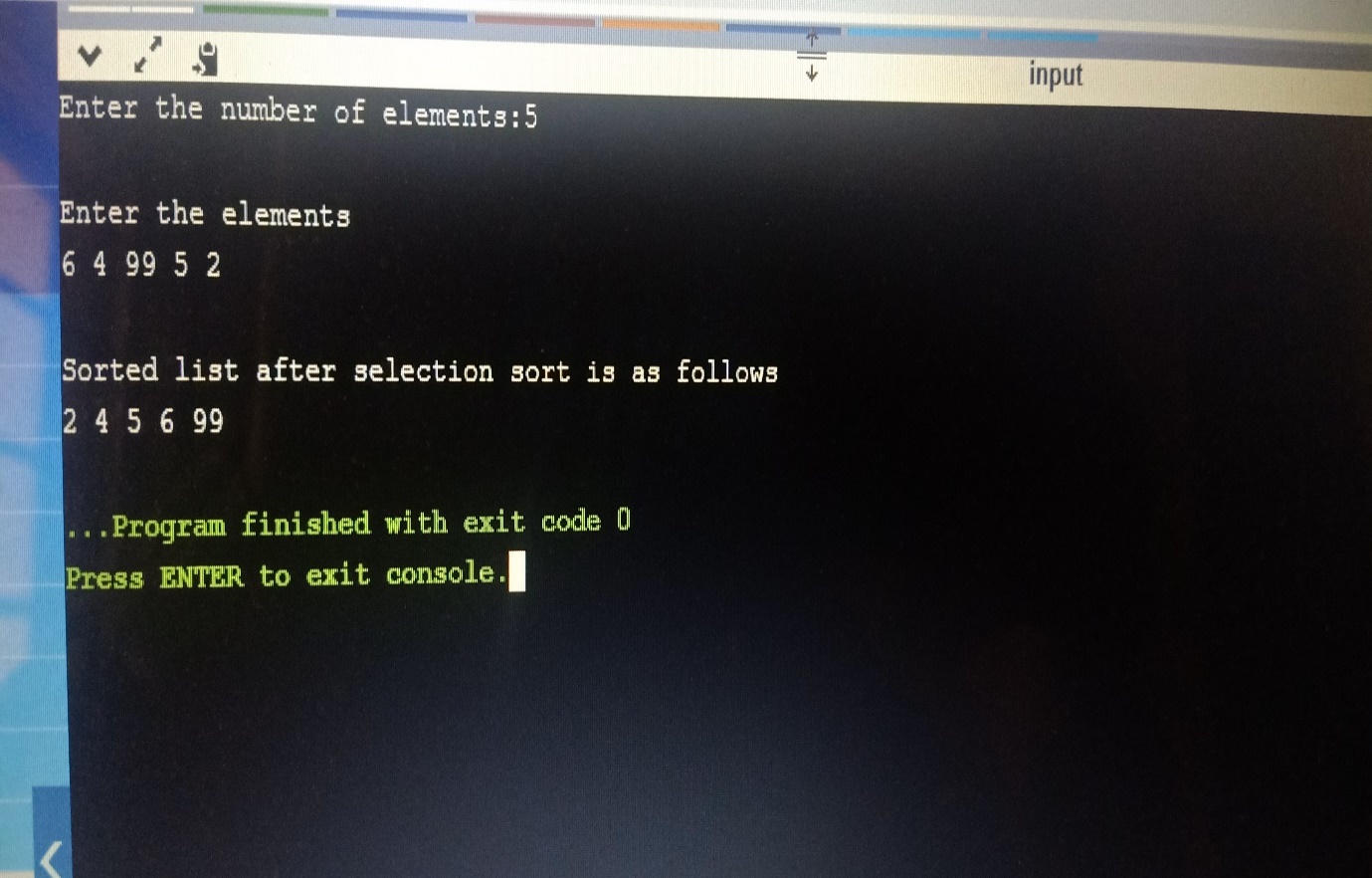
printf("%d ",a[i]);

}

return 0;

}

**Output:**

****

**Experiment-IV**

**Aim:** Write a program to implement Linear search.

**Program:**

#include <stdio.h>

int main()

{

int a[20],n,x,i,flag=0;

printf("Enter number of Elements:");

scanf("%d",&n);

printf("\nEnter elements of the array\n");

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

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

printf("\nEnter element to search:");

scanf("%d",&x);

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

{

if(a[i]==x)

{

flag=1;

break;

}

}

if(flag)

printf("\nElement is found at position %d",i+1);

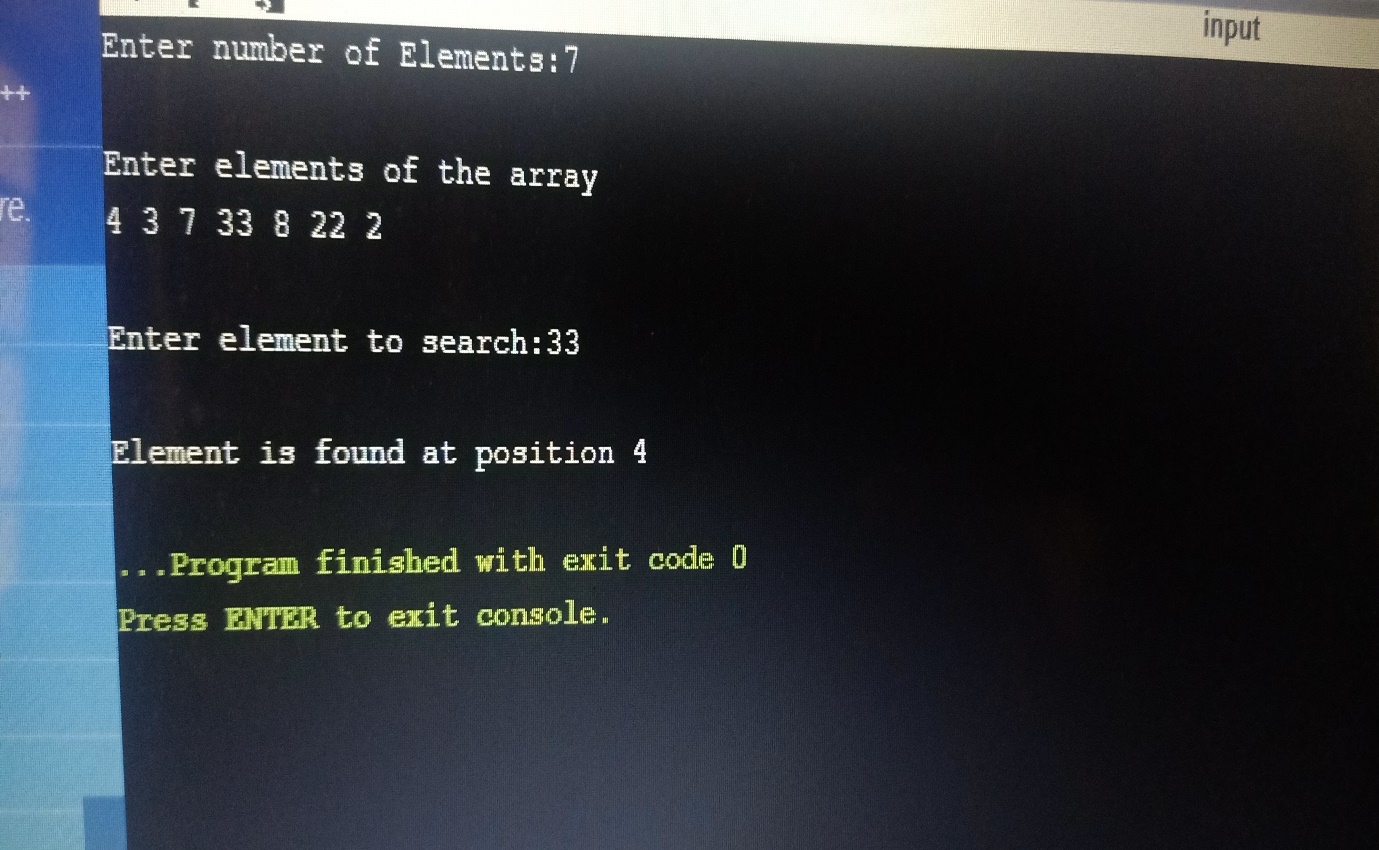
else

printf("\nElement not found");

return 0;

}

**Output:**

****

**Experiment-V**

**Aim:** Write a program to implement Binary Search.

**Program:**

#include <stdio.h>

int main()

{

int search(int [],int,int);

int n,i,a[100],e,res;

printf("Enter number of Elements:");

scanf("%d",&n);

printf("\nEnter Elements of Array in Ascending order\n");

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

{

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

}

printf("\nEnter element to search:");

scanf("%d",&e);

res=search(a,n,e);

if(res!=-1)

printf("\nElement found at position %d",res+1);

else

printf("\nElement is not found....!!!");

return 0;

}

int search(int a[],int n,int e)

{

int f,l,m;

f=0;

l=n-1;

while(f<=l)

{

m=(f+l)/2;

if(e==a[m])

return(m);

else

if(e>a[m])

f=m+1;

else

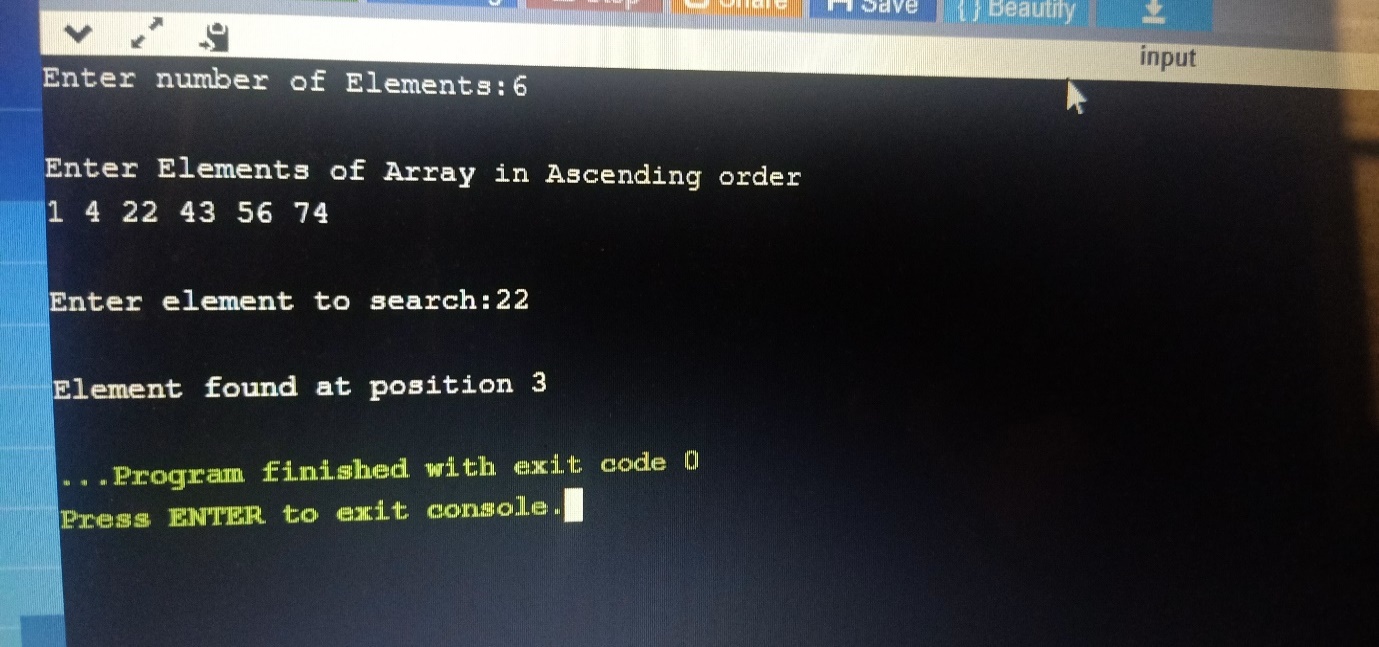
l=m-1;

}

return -1;

}

**Output:**

****

**Experiment-VI**

**Aim:** Write a program to implement Quick Sort.

**Program:**

#include <stdio.h>

void swap(int\* a, int\* b)

{

int t = \*a;

\*a = \*b;

\*b = t;

}

int partition (int arr[], int low, int high)

{

int pivot = arr[high];

int i = (low - 1);

for (int j = low; j <= high- 1; j++)

{

if (arr[j] <= pivot)

{

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i + 1], &arr[high]);

return (i + 1);

}

void quickSort(int arr[], int low, int high)

{

if (low < high)

{

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

void printArray(int arr[], int size)

{

int i;

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

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

}

int main()

{

int n, i;

printf("\nEnter the number of data element to be sorted: ");

scanf("%d",&n);

int arr[n];

printf("Enter element:");

for(i = 0; i < n; i++) {

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

}

quickSort(arr, 0, n-1);

printf("Sorted array:");

printArray(arr, n);

return 0;

}

**Output:**

****

**Experiment-VII**

**Aim:** Write a program to implement Merge Sort.

**Program:**

#include <stdio.h>

void merge(int arr[], int l, int m, int r)

{

int n1 = m - l + 1;

int n2 = r - m;

int L[n1], R[n2];

for (int i = 0; i < n1; i++)

L[i] = arr[l + i];

for (int j = 0; j < n2; j++)

R[j] = arr[m + 1 + j];

int i = 0;

int j = 0;

int k = l;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

}

else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(int arr[],int l,int r){

if(l>=r){

return;

}

int m =l+ (r-l)/2;

mergeSort(arr,l,m);

mergeSort(arr,m+1,r);

merge(arr,l,m,r);

}

void printArray(int A[], int size)

{

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

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

}

int main()

{

int n, i;

printf("\nEnter the number of data element to be sorted: ");

scanf("%d",&n);

int arr[n];

printf("Enter element:");

for(i = 0; i < n; i++) {

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

}

mergeSort(arr, 0, n-1);

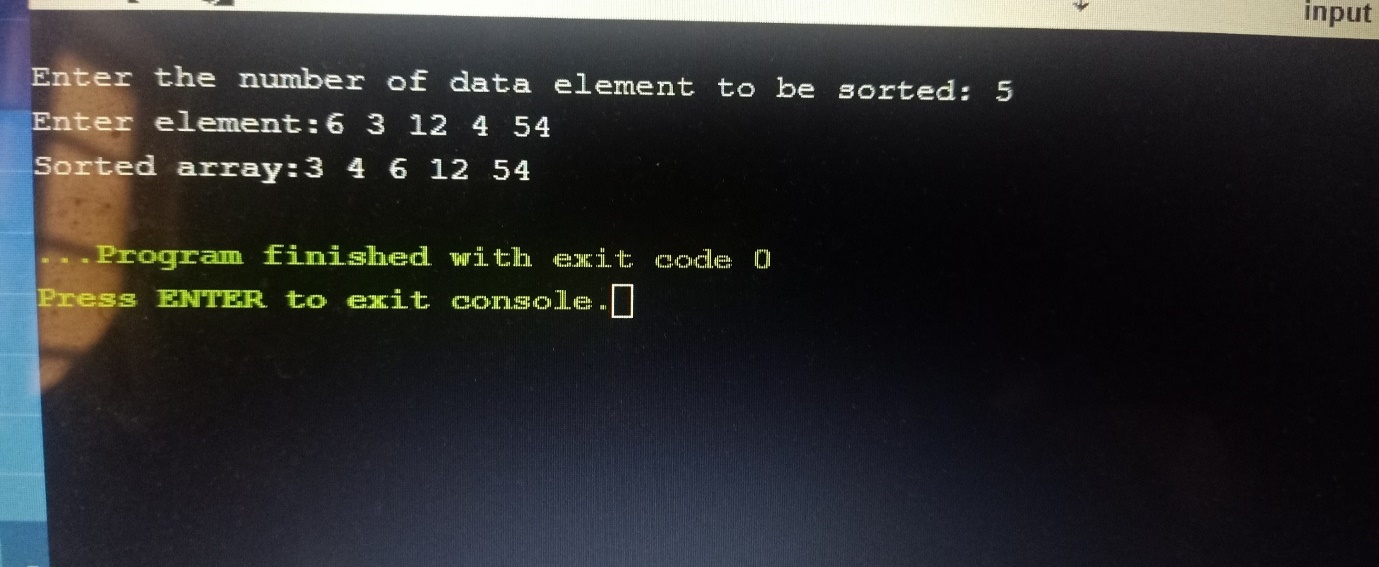
printf("Sorted array:");

printArray(arr, n);

return 0;

}

**Output:**

****

**Experiment-VIII**

**Aim:** Write a program to implement heap sort.

**Program:**

#include <stdio.h>

void swap(int \*a,int \*b){

int temp;

temp=\*a;

\*a=\*b;

\*b=temp;

}

void heapify(int arr[],int n,int i){

int max = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < n && arr[l] > arr[max]){

max = l;

}

if (r < n && arr[r] > arr[max]){

max = r;

}

if (max != i) {

swap(&arr[i], &arr[max]);

heapify(arr, n, max);

}

}

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--) {

swap(&arr[0], &arr[i]);

heapify(arr, i, 0);

}

}

int main()

{

int n;

printf("Enter the number of elements in array:");

scanf("%d",&n);

int arr[n];

printf("Enter the elements in an array:");

for(int i=0;i<n;i++){

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

}

heapSort(arr,n);

printf("Sorted Array after heap Sort is:");

for(int i=0;i<n;i++){

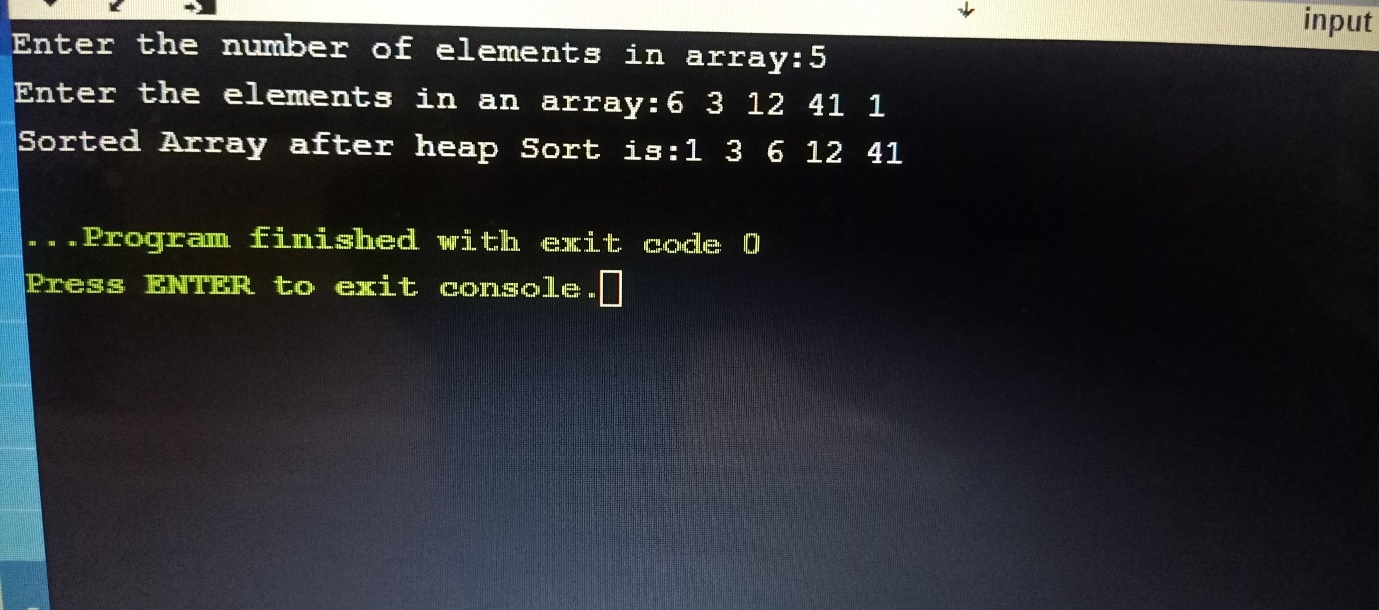
printf("%d ",arr[i]);

}

return 0;

}

**Output:**

****

**Experiment-IX**

**Aim:** Write a program to implement radix sort.

**Program:**

#include <stdio.h>

int getMaximum(int arr[],int n){

int max=arr[0];

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

if(arr[i]>max){

max=arr[i];

}

}

return max;

}

void radixSort(int arr[],int n){

int max=getMaximum(arr,n);

for (int exp = 1; max / exp > 0; exp \*= 10){

int output[n];

int i, count[10] = { 0 };

for (i = 0; i < n; i++){

count[(arr[i] / exp) % 10]++;

}

for (i = 1; i < 10; i++) {

count[i] += count[i - 1];

}

for (i = n - 1; i >= 0; i--) {

output[count[(arr[i] / exp) % 10] - 1] = arr[i];

count[(arr[i] / exp) % 10]--;

}

for (i = 0; i < n; i++) {

arr[i] = output[i];

}

}

}

int main()

{

int n;

printf("Enter the number of elements in array:");

scanf("%d",&n);

int arr[n];

printf("Enter elements of array:");

for(int i=0;i<n;i++){

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

}

radixSort(arr,n);

printf("Sorted Array after radix sort is:");

for(int i=0;i<n;i++){

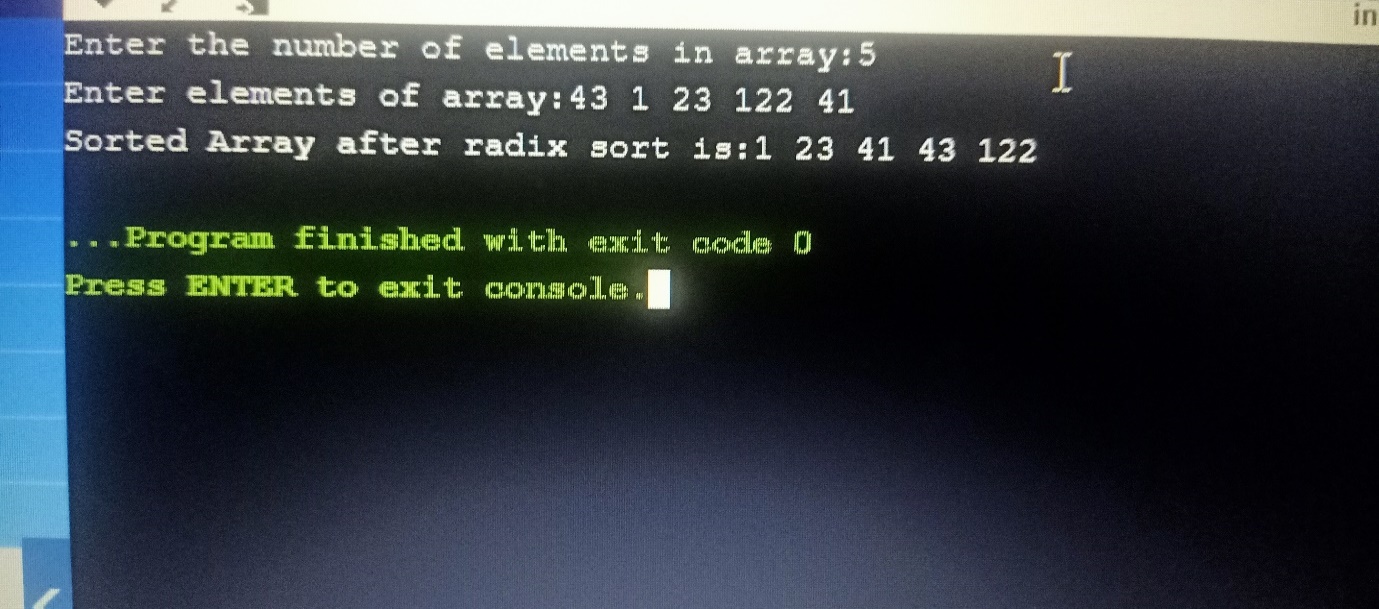
printf("%d ",arr[i]);

}

return 0;

}

**Output:**

****

**Experiment-X**

**Aim:** Write a program to implement Binary Search Recursive.

**Program:**

#include <stdio.h>

int binarySearch(int arr[], int low, int high, int search)

{

if (high >= low)

{

int mid = low + (high - low)/2;

if (arr[mid] == search){

return mid;

}

if (arr[mid] > search){

return binarySearch(arr, low, mid-1, search);

}

else{

return binarySearch(arr, mid+1, high, search);

}

}

return -1;

}

int main()

{

int n,search;

printf("Enter the number of elements in array:");

scanf("%d",&n);

int a[n];

printf("Enter the elements of array:");

for(int i=0;i<n;i++){

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

}

printf("Enter the element to search:");

scanf("%d",&search);

for(int i=0;i<n;i++){

for(int j=0;j<n-1;j++){

if(a[j]>a[j+1]){

int temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

printf("Array after Sorting:");

for(int i=0;i<n;i++){

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

}

int found=binarySearch(a,0,n-1,search);

if(found==-1){

printf("\nElement not found");

}

else{

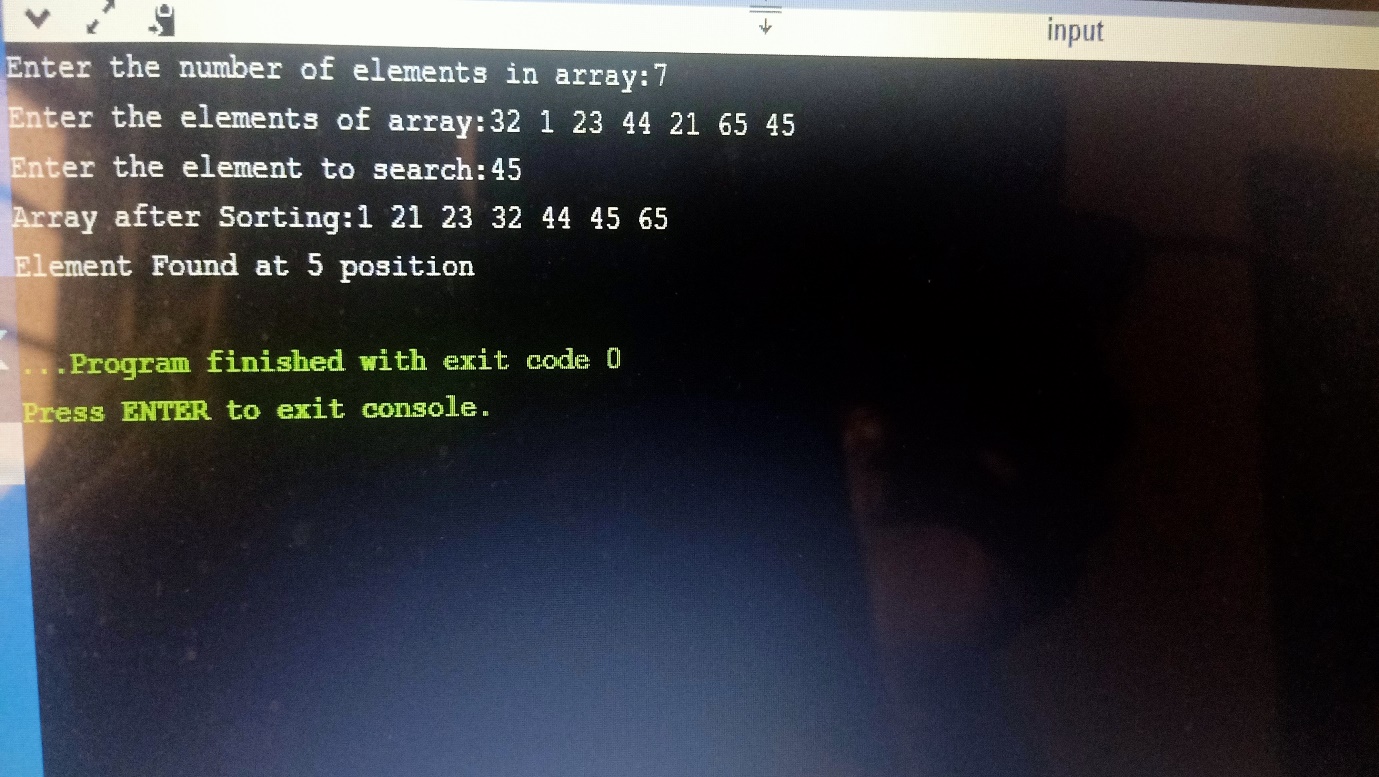
printf("\nElement Found at %d position",found);

}

return 0;

}

**Output:**

****

**Experiment-XI**

**Aim:** Write a program to implement Strassen’s matrix multiplication.

**Program:**

#include<stdio.h>

int main(){

int a[2][2],b[2][2],c[2][2],i,j;

int m1,m2,m3,m4,m5,m6,m7;

printf("Enter the 4 elements of first matrix: ");

for(i=0;i<2;i++)

for(j=0;j<2;j++)

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

printf("Enter the 4 elements of second matrix: ");

for(i=0;i<2;i++)

for(j=0;j<2;j++)

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

printf("\nThe first matrix is");

for(i=0;i<2;i++){

printf("\n");

for(j=0;j<2;j++)

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

}

printf("\nThe second matrix is");

for(i=0;i<2;i++){

printf("\n");

for(j=0;j<2;j++)

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

}

m1= (a[0][0] + a[1][1])\*(b[0][0]+b[1][1]);

m2= (a[1][0]+a[1][1])\*b[0][0];

m3= a[0][0]\*(b[0][1]-b[1][1]);

m4= a[1][1]\*(b[1][0]-b[0][0]);

m5= (a[0][0]+a[0][1])\*b[1][1];

m6= (a[1][0]-a[0][0])\*(b[0][0]+b[0][1]);

m7= (a[0][1]-a[1][1])\*(b[1][0]+b[1][1]);

c[0][0]=m1+m4-m5+m7;

c[0][1]=m3+m5;

c[1][0]=m2+m4;

c[1][1]=m1-m2+m3+m6;

printf("\nAfter multiplication");

for(i=0;i<2;i++){

printf("\n");

for(j=0;j<2;j++)

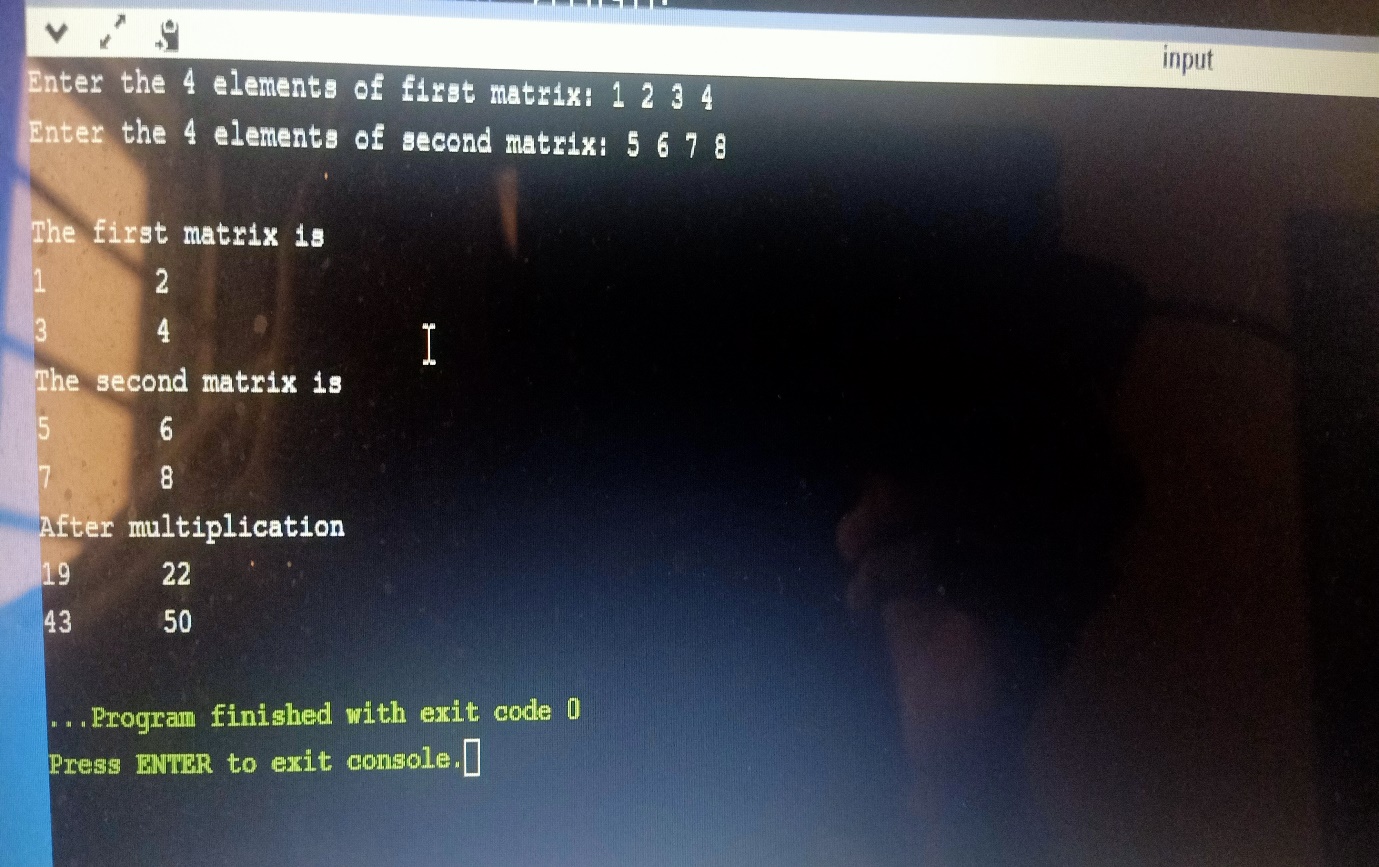
printf("%d\t",c[i][j]);

}

return 0;

}

**Output:**

****

**Experiment-XII**

**Aim:** Write a program to implement optimal merge pattern.

**Program:**

#include <stdio.h>

int main()

{

int n;

printf("Enter number of files to merge:");

scanf("%d",&n);

int arr[n],c[n-1];

printf("Enter the sorted elements for optimal merge pattern:");

for(int i=0;i<n;i++){

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

}

int i=0,k=0,l=0;

c[k]=arr[i]+arr[i+1];

i=2;

while(i<n){

k++;

if(c[k-1]+arr[i]<=arr[i]+arr[i+1]){

c[k]=c[k-1]+arr[i];

}

else{

c[k]=arr[i]+arr[i+1];

i=i+2;

while(i<n)

{

k++;

if(c[k-1]<=c[k-2])

{

c[k]=c[k-1]+arr[i];

}

else

{

c[k]=c[k-2]+arr[i];

}

i++;

}

}

i++;

}

k++;

c[k]=c[k-1]+c[k-2];

printf("The optimal sum are:");

for(k=0;k<n-1;k++)

{

printf("%d ",c[k]);

}

for(k=0;k<n-1;k++)

{

l=l+c[k];

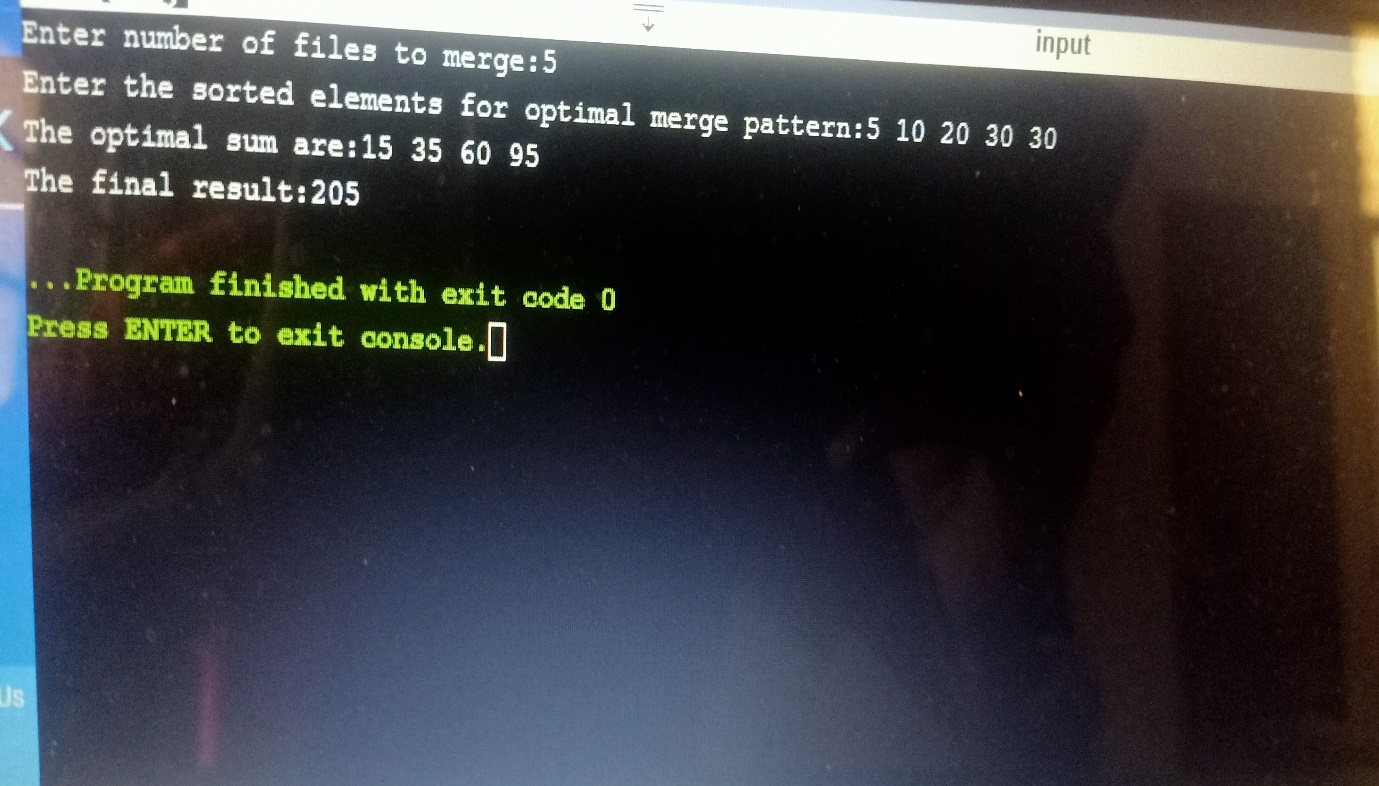
}

printf("\nThe final result:%d",l);

return 0;

}

**Output:**

****

**Experiment-XIII**

**Aim:** program to implement fractional knapsack problem in c.

**Program:**

# include<stdio.h>

void knapsack(int n, float weight[], float value[], float capacity) {

float x[20], t = 0;

int i, j, u;

u = capacity;

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

x[i] = 0.0;

for (i = 0; i < n; i++) {

if (weight[i] > u)

break;

else {

x[i] = 1.0;

t = t + value[i];

u = u - weight[i];

}

}

if (i < n)

x[i] = u / weight[i];

t = t + (x[i] \* value[i]);

printf("\nThe result is:- ");

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

printf("%f\t", x[i]);

printf("\nMaximum profit is:- %f", t);

}

int main() {

float weight[20], value[20], capacity;

int num, i, j;

float ratio[20], temp;

printf("\nEnter the no. of objects:- ");

scanf("%d", &num);

printf("\nEnter the wts and profits of each object:- ");

for (i = 0; i < num; i++) {

scanf("%f %f", &weight[i], &value[i]);

}

printf("\nEnter the capacity of knapsack:- ");

scanf("%f", &capacity);

for (i = 0; i < num; i++) {

ratio[i] = value[i] / weight[i];

}

for (i = 0; i < num; i++) {

for (j = i + 1; j < num; j++) {

if (ratio[i] < ratio[j]) {

temp = ratio[j];

ratio[j] = ratio[i];

ratio[i] = temp;

temp = weight[j];

weight[j] = weight[i];

weight[i] = temp;

temp = value[j];

value[j] = value[i];

value[i] = temp;

}

}

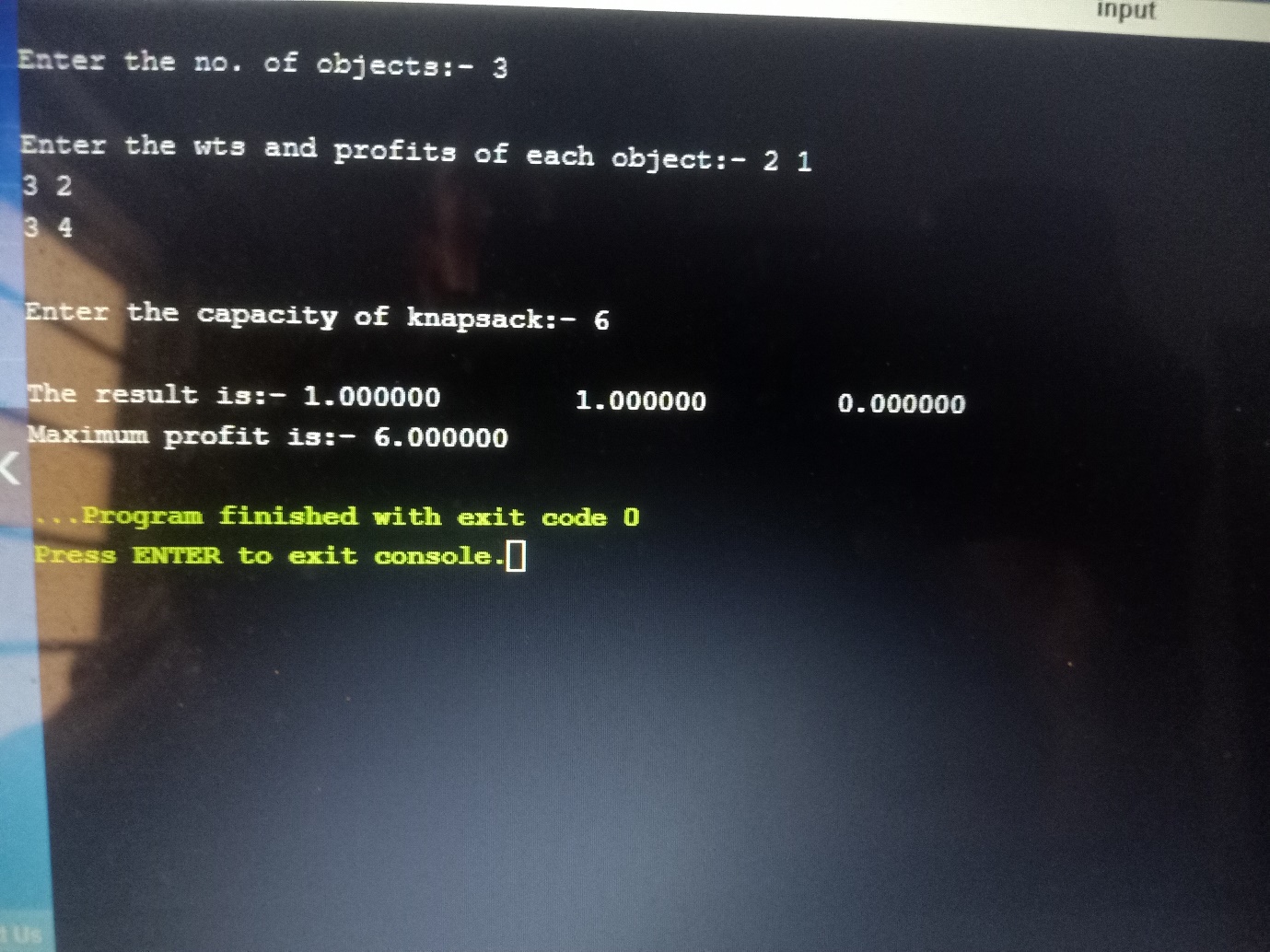
}

knapsack(num, weight, value, capacity);

return(0);

}

**Output:**

****

**Experiment-XIV**

**Aim:** Program to implement dijkstras algorithm in c.

**Program:**

#include <stdio.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("Enter 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;

}

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];

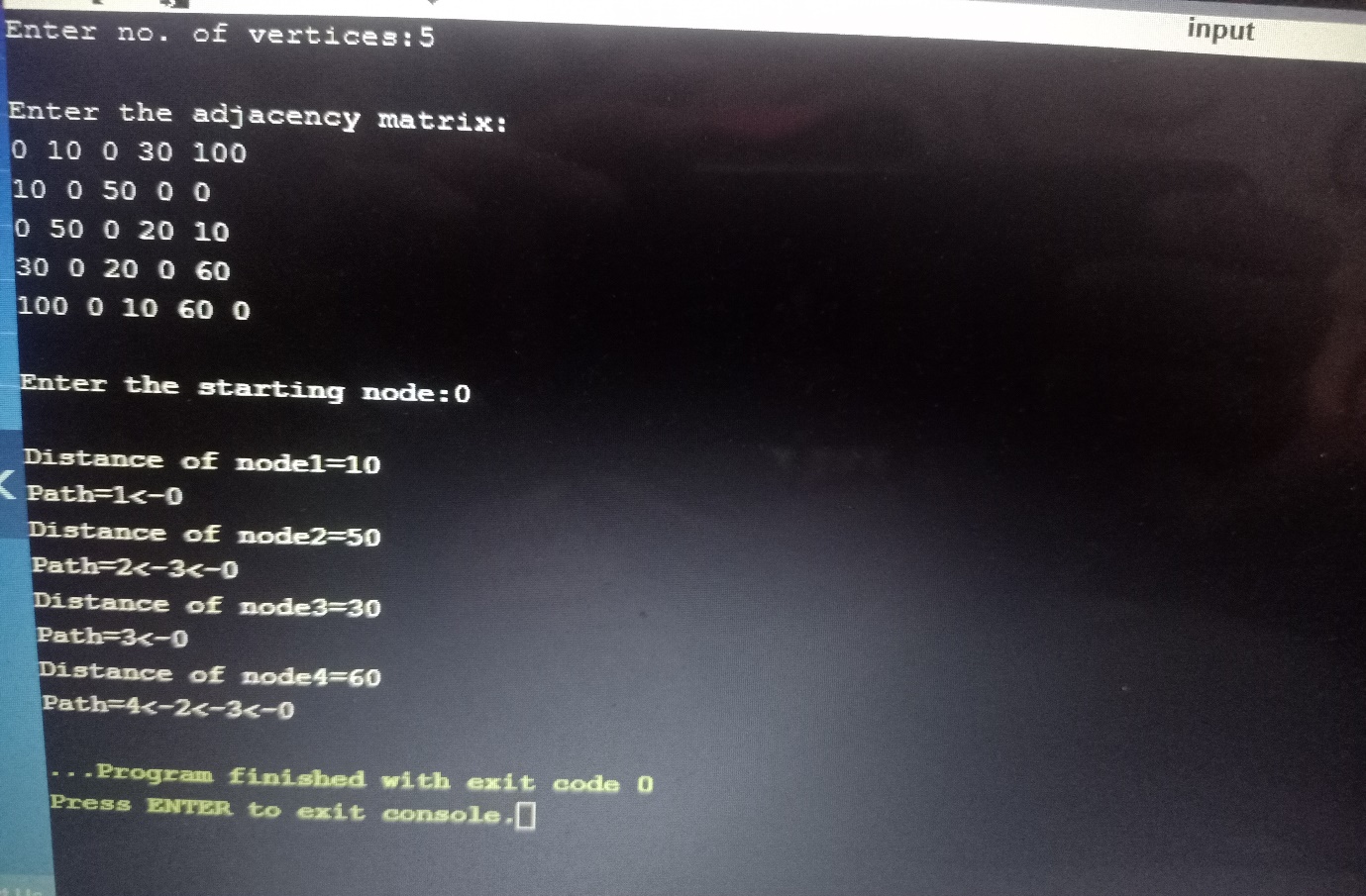
printf("<-%d",j);

}while(j!=startnode);

}

}

**Output:**

****

**Experiment-XV**

**Aim:** Program to implement job sequencing with deadline in c.

**Program:**

#include <stdio.h>

int n,i,j,k,t;

int check(int s[],int p)

{ int ptr=0;

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

{if(s[i]==p)

ptr++;

}

if(ptr==0)

return 1;

else

return 0;

}

int main()

{

printf("Enter the no of jobs:");

scanf("%d",&n);

int slot[n],profit[n],deadline[n],max\_profit=0;

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

{

printf("Enter the profit of job #%d:",i+1);

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

printf("Enter the deadline of job #%d:",i+1);

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

}

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

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

if(profit[i]<profit[j])

{

t=profit[i];

profit[i]=profit[j];

profit[j]=t;

t=deadline[i];

deadline[i]=deadline[j];

deadline[j]=t;

}

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

slot[i]=0;

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

for(j=deadline[i];j>0;j--)

{

if(check(slot,j)==1)

{

slot[i]=j;

break;

}

}

printf("\n INDEX PROFIT DEADLINE SLOT ALLOTTED ");

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

{

if(slot[i]>0){

printf("\n\n %d %d %d [%d - %d]", i+1,profit[i],deadline[i],(slot[i]-1),slot[i]);

max\_profit=max\_profit+profit[i];

}

else{

printf("\n\n %d %d %d REJECTED", i+1,profit[i],deadline[i]);

}

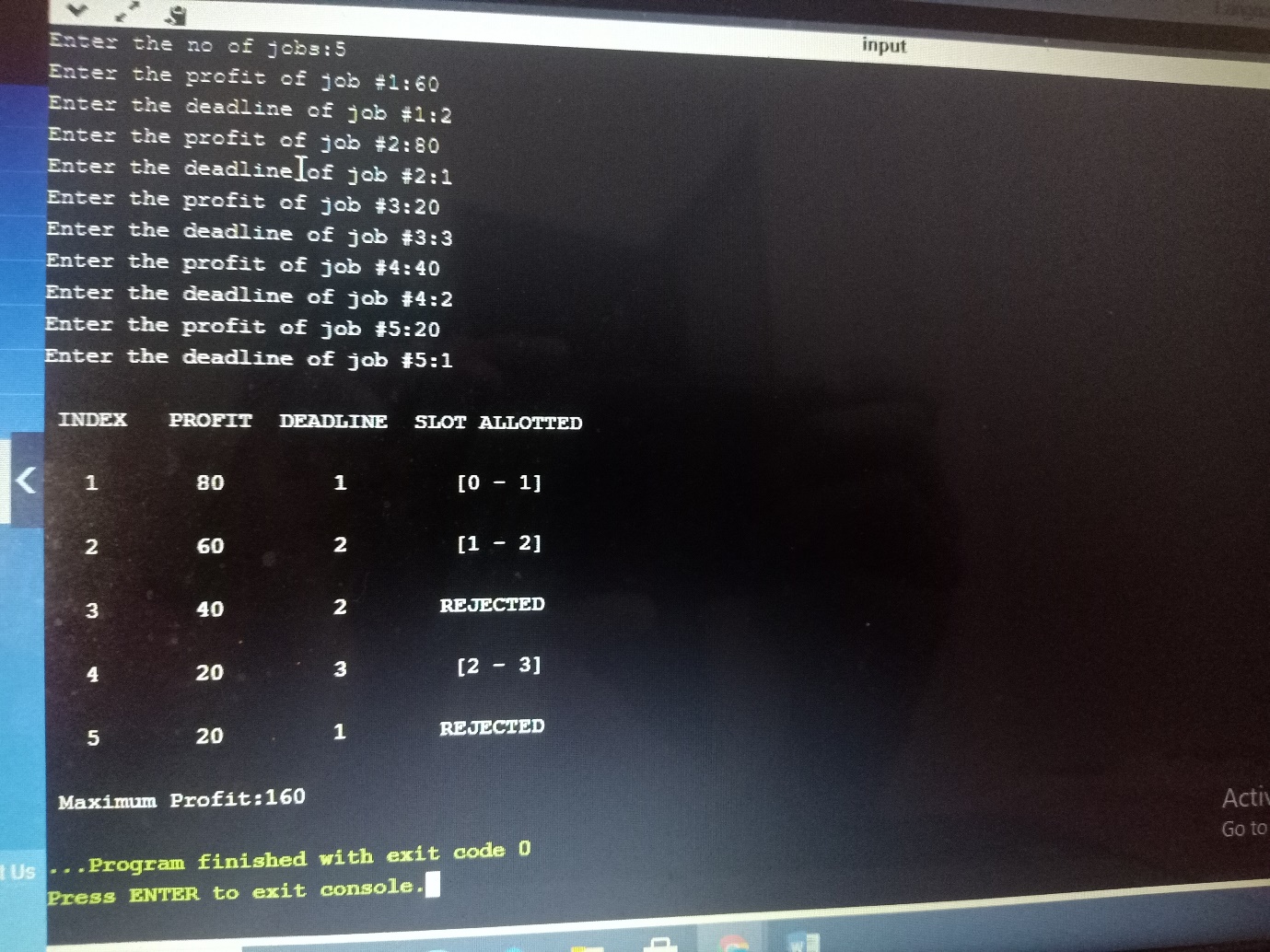
}

printf("\n\n Maximum Profit:%d",max\_profit);

return 0;

}

**Output:**

****

**Experiment-XVI**

**Aim:** Program to implement Kruskal’s algorithm in c.

**Program:**

#include <stdio.h>

int i,j,k,a,b,u,v,n,ne=1;

int min,mincost=0,parent[9];

int findNode(int i)

{

while(parent[i])

i=parent[i];

return i;

}

int union1(int i,int j)

{

if(i!=j)

{

parent[j]=i;

return 1;

}

return 0;

}

int main()

{

printf("\nEnter the no. of vertices:");

scanf("%d",&n);

printf("\nEnter the adjacency matrix:\n");

int cost[n][n];

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

{

for(j=1;j<=n;j++)

{

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

if(cost[i][j]==0)

cost[i][j]=999;

}

}

printf("The edges of Minimum Spanning Tree are\n");

while(ne < n)

{

for(i=1,min=999;i<=n;i++)

{

for(j=1;j <= n;j++)

{

if(cost[i][j] < min)

{

min=cost[i][j];

a=u=i;

b=v=j;

}

}

}

u=findNode(u);

v=findNode(v);

if(union1(u,v))

{

printf("%d edge (%d,%d) =%d\n",ne++,a,b,min);

mincost +=min;

}

cost[a][b]=cost[b][a]=999;

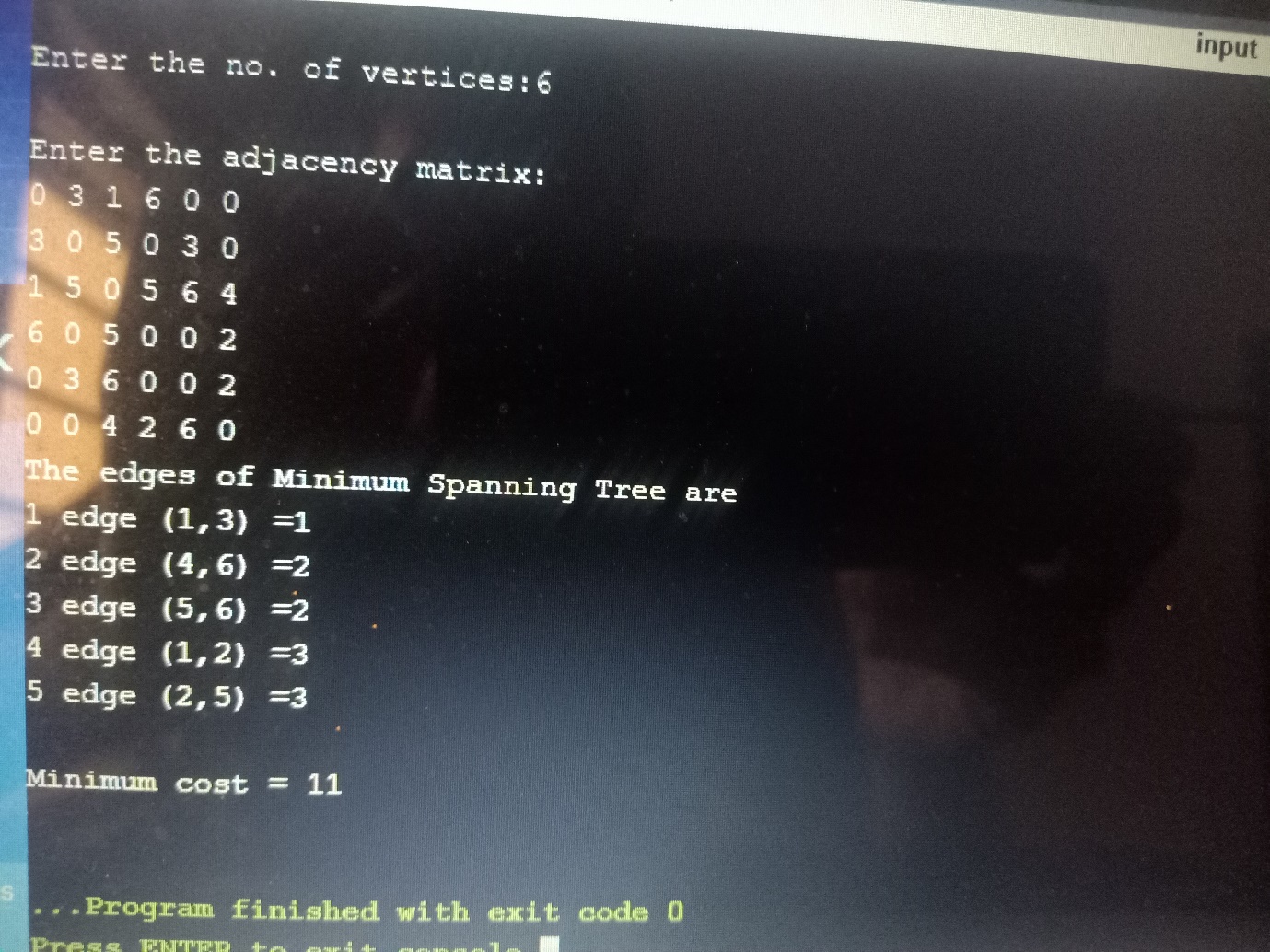
}

printf("\nMinimum cost = %d\n",mincost);

return 0;

}

**Output:**

****

**Experiment-XVII**

**Aim:** Program to implement Prim’s algorithm in c.

**Program:**

#include<stdio.h>

#include<conio.h>

int a,b,u,v,n,i,j,ne=1;

int visited[10]={0},min,mincost=0,cost[10][10];

int main()

{

printf("\nEnter the number of nodes:");

scanf("%d",&n);

printf("\nEnter the adjacency matrix:\n");

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

for(j=1;j<=n;j++)

{

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

if(cost[i][j]==0){

cost[i][j]=999;

}

}

visited[1]=1;

printf("\n");

while(ne < n)

{

for(i=1,min=999;i<=n;i++)

for(j=1;j<=n;j++)

if(cost[i][j]< min)

if(visited[i]!=0)

{

min=cost[i][j];

a=u=i;

b=v=j;

}

if(visited[u]==0 || visited[v]==0)

{

printf("\n Edge %d:(%d %d) cost:%d",ne++,a,b,min);

mincost+=min;

visited[b]=1;

}

cost[a][b]=cost[b][a]=999;

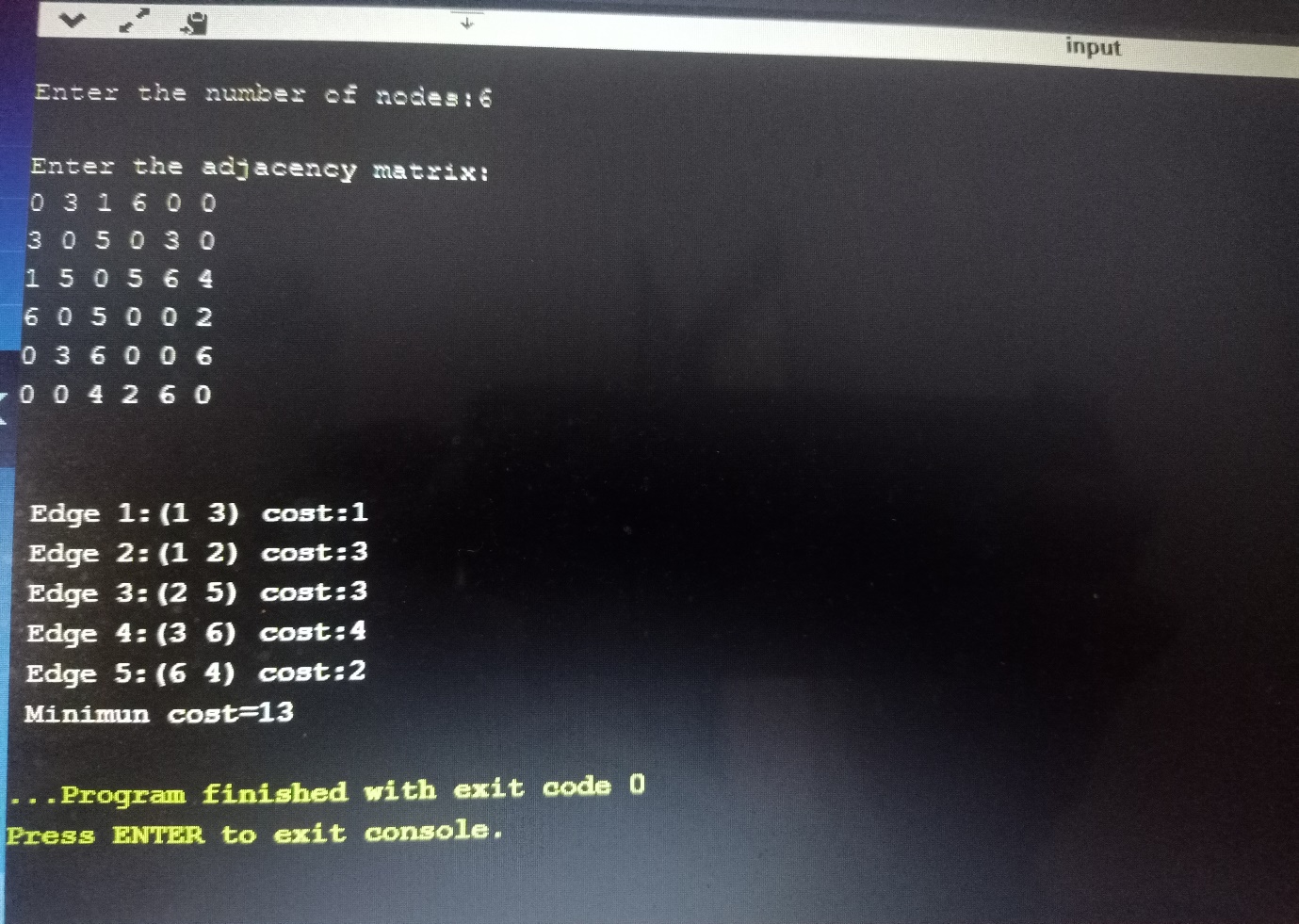
}

printf("\n Minimun cost=%d",mincost);

return 0;

}

**Output:**

****