AMITY UNIVERSITY MADHYA PRADESH

Practical File

of

DESIGN AND ANALYSIS OF ALGORITHMS LAB

CSE-323

SUBMITTED TO:-

MR. DEVENDRA KUMAR MISHRA

Asst. Professor

CSE-ASET

SUBMITTED BY:-

KSHITIJ AGRAWAL

B.Tech [CSE]

3RD SEMESTER

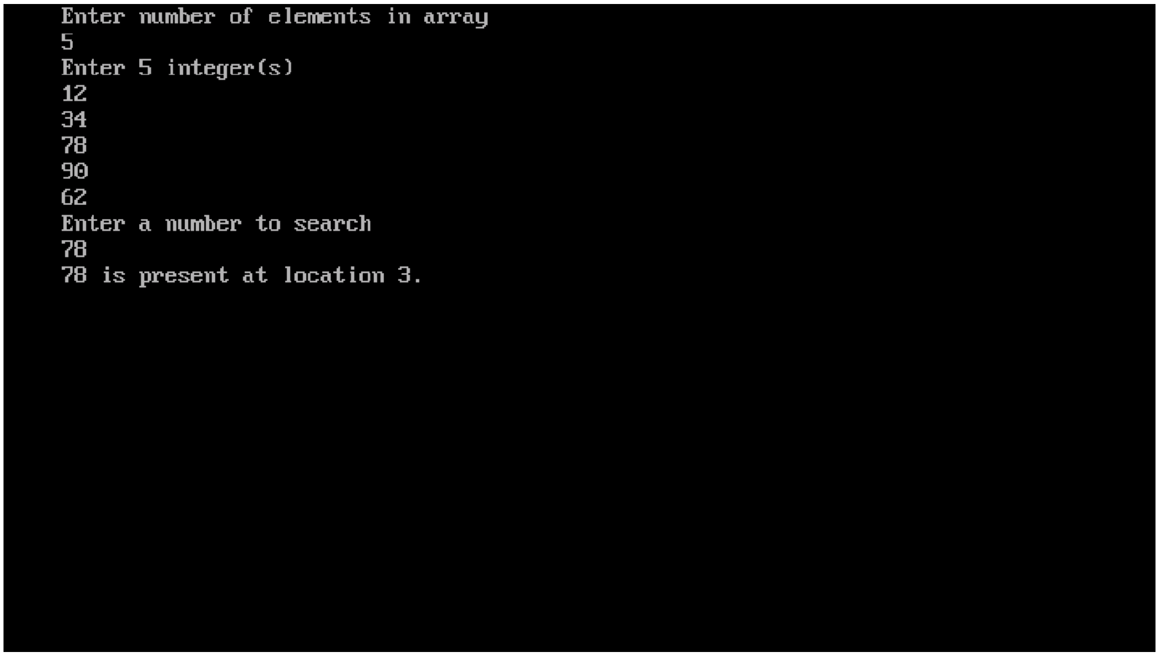
INDEX

Ques – 1:- Program for linear search.

Code :-

```
#include <stdio.h>
#include<conio.h>
void main()
{
  clrscr();
  int a[50],search,i,n;
  printf("Enter number of elements in array\n");
  scanf("%d",&n);
  printf("Enter %d integer(s)\n", n);
  for (i=0;i<n;i++)
    scanf("%d",&a[i]);
  printf("Enter a number to search\n");
  scanf("%d",&search);
  for (i=0;i<n;i++)
  {
    if (a[i] == search)
    {
      printf("%d is present at location %d.\n", search, i+1);
      break;
    }
  }
  if (i == n)
    printf("%d isn't present in the array.\n", search);
  getch();
}
```

Output is :-



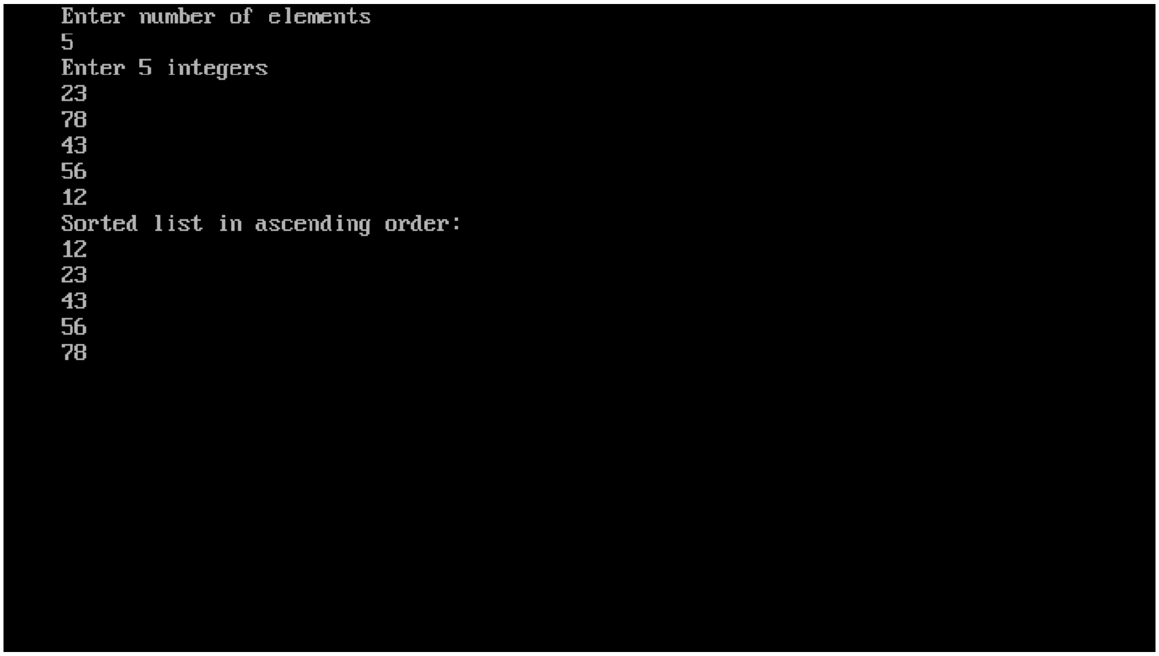
```
Enter number of elements in array
5
Enter 5 integer(s)
12
34
78
90
62
Enter a number to search
78
78 is present at location 3.
```

Ques 2:- Program for Bubble sort .

Code :-

```
#include <stdio.h>
#include <conio.h>
void main()
{
clrscr();
  int array[100], n, i, j, swap;
  printf("Enter number of elements\n");
  scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (i=0;i<n;i++)
    scanf("%d", &array[i]);
  for (i=0;i<n-1;i++)
  {
    for (j=0;j<n-i-1;j++)
    {
      if (array[j]>array[j+1])
      {
        swap       = array[j];
        array[j]   = array[j+1];
        array[j+1] = swap;
      }
    }
  }
  printf("Sorted list in ascending order:\n");
  for (i=0;i<n;i++)
     printf("%d\n", array[i]);
  getch();
}
```

Output is :-



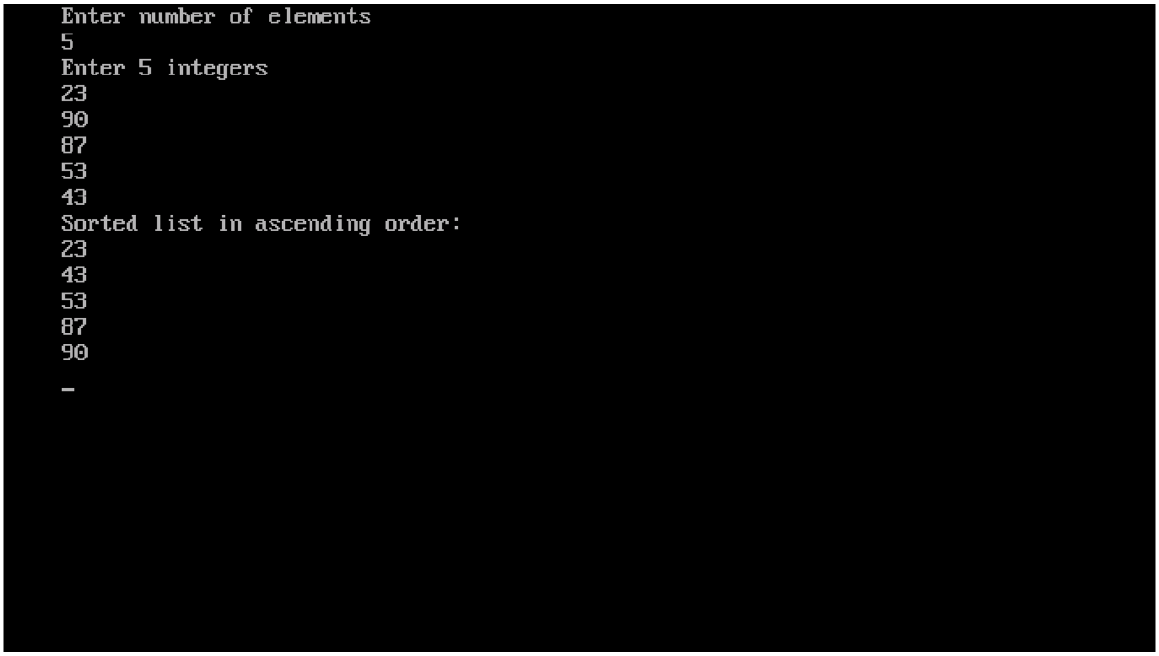
```
Enter number of elements
5
Enter 5 integers
23
78
43
56
12
Sorted list in ascending order:
12
23
43
56
78
```

Ques 3:- Program for Selection sort .

Code :-

```
#include <stdio.h>
#include <conio.h>
void main()
{
    clrscr();
  int array[100], n, i, j, p, t;
  printf("Enter number of elements\n");
  scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (i=0;i<n;i++)
    scanf("%d", &array[i]);
  for (i=0;i<(n-1);i++)
  {
      p=i;
    for (j=i+1;j<n;j++)
    {
      if (array[p]>array[j])
      p=j;
    }
    if (p!=i)
      {
        t       = array[i];
        array[i]   = array[p];
        array[p] = t;
      }
  }
  printf("Sorted list in ascending order:\n");
  for (i=0;i<n;i++)
     printf("%d\n", array[i]);
  getch();
}
```

Output is :-



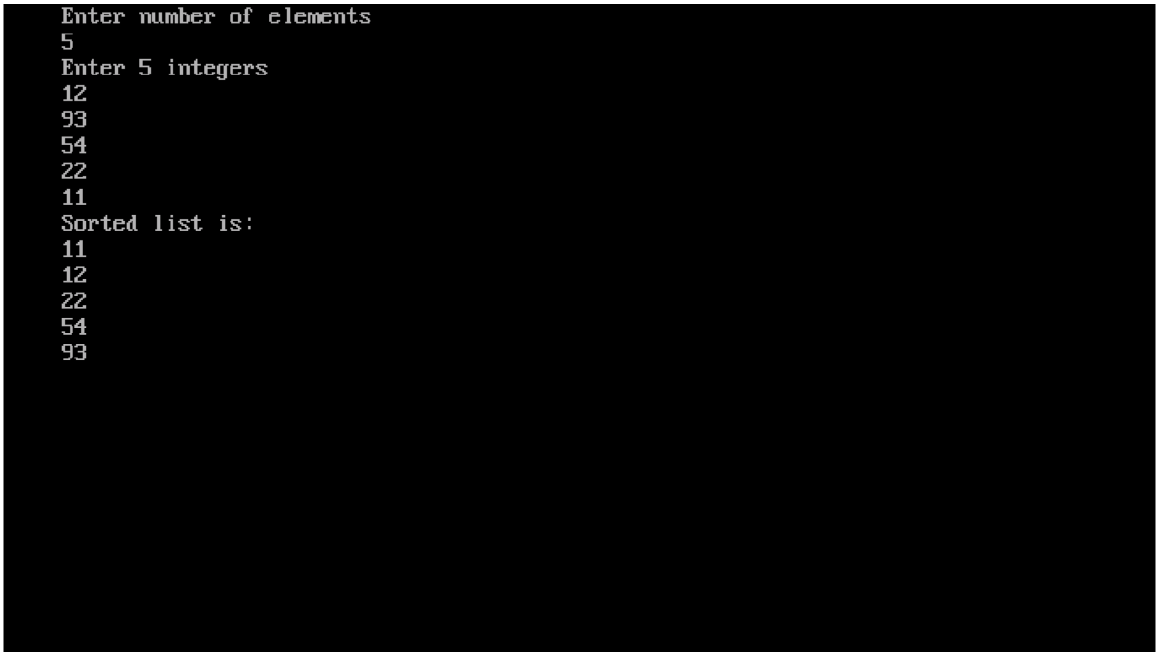
```
Enter number of elements
5
Enter 5 integers
23
90
87
53
43
Sorted list in ascending order:
23
43
53
87
90

_
```

Ques 4 :- Program for Insertion sort .

Code :-

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
  int n,array[100],i,j,k,temp=0;
  printf("Enter number of elements\n");
  scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (i=0;i<n;i++)
    scanf("%d", &array[i]);
  for (i=1;i<=n-1;i++)
  {
    k = array[i];
    for (j=i-1;j>=0;j--)
    {
      if (array[j]>k)
      {
        array[j+1]=array[j];
        temp=1;
      }
      else
        break;
    }
    if (temp)
      array[j+1] = k;
  }
  printf("Sorted list is:\n");
  for (i=0;i<=n-1;i++)
  {
    printf("%d\n", array[i]);
  }
  getch();
}
```

Output is :-



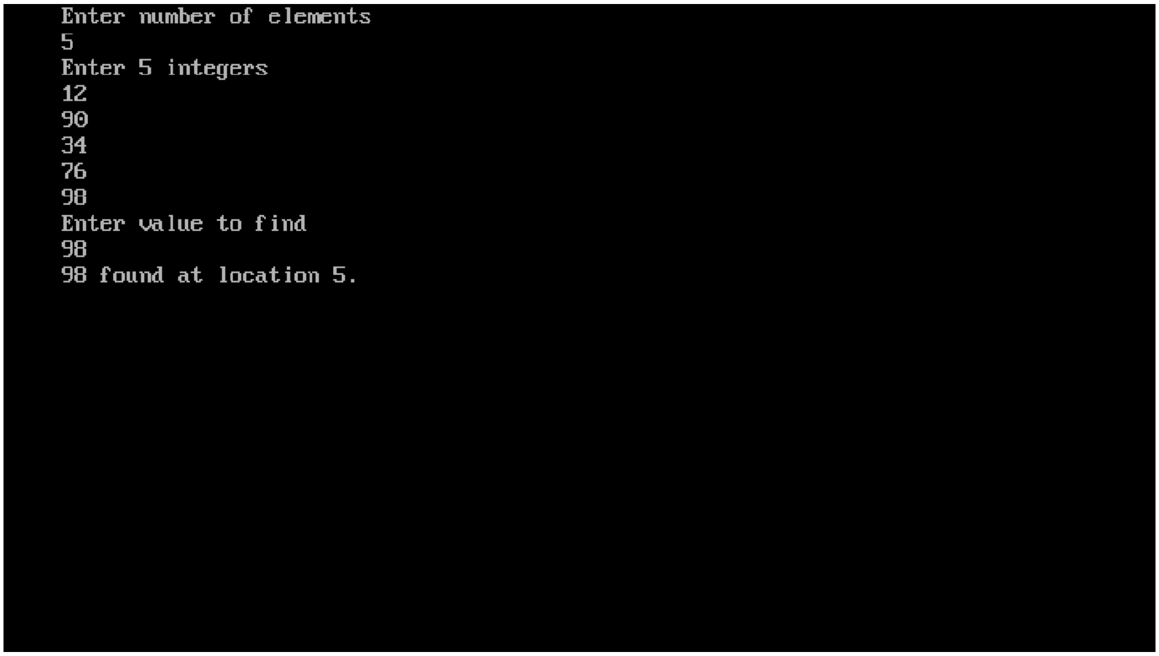
```
Enter number of elements
5
Enter 5 integers
12
93
54
22
11
Sorted list is:
11
12
22
54
93
```

Ques 5:- Program for Binary Search .

Code :-

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
  int n,i, first, last, middle, search, array[1000];
  printf("Enter number of elements\n");
  scanf("%d",&n);
  printf("Enter %d integers\n", n);
  for (i=0;i<n;i++)
    scanf("%d",&array[i]);
  printf("Enter value to find\n");
  scanf("%d",&search);
  first = 0;
  last = n - 1;
  middle = (first+last)/2;

  while (first <= last)
  {
    if (array[middle] < search)
      first = middle + 1;
    else if (array[middle] == search)
    {
      printf("%d found at location %d.\n", search, middle+1);
      break;
    }
    else
      last = middle - 1;
    middle = (first + last)/2;
  }
  if (first > last)
    printf("Not found! %d isn't present in the list.\n", search);
  getch();
}
```

Output is :-



```
Enter number of elements
5
Enter 5 integers
12
90
34
76
98
Enter value to find
98
98 found at location 5.
```

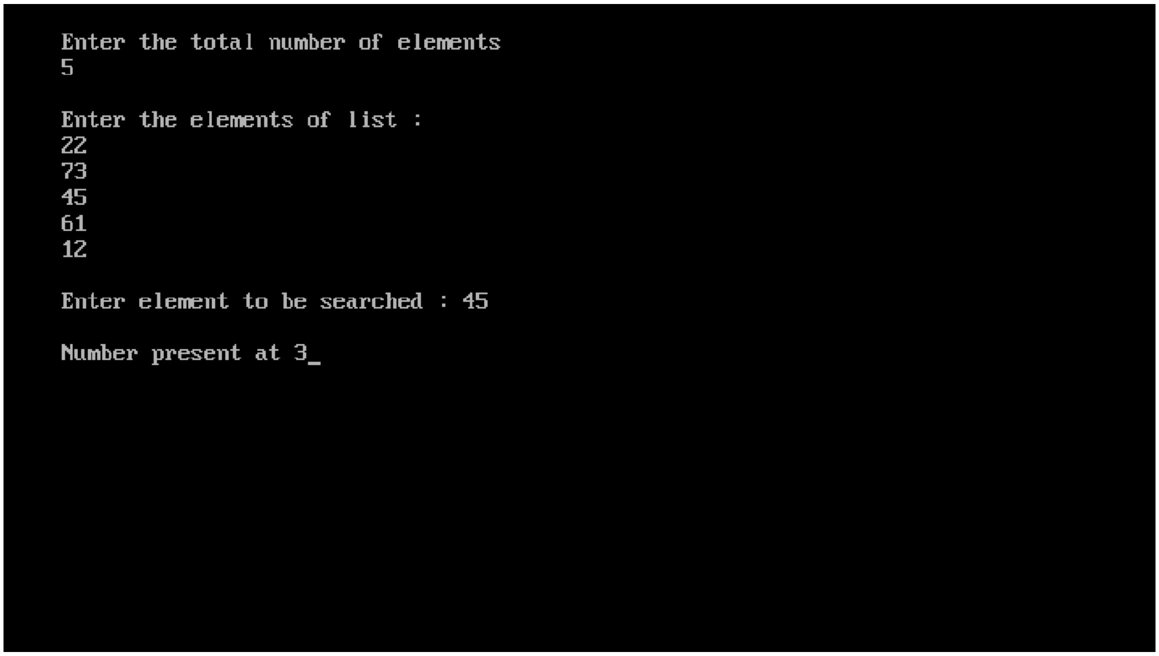
Ques 6:- Program for Binary Search using recursion .

Code:-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define size 10

int binsearch(int[], int, int, int);

int main()
 {
   clrscr();
   int num, i, key, position;

   int low, high, list[size];
   printf("\nEnter the total number of elements");
   scanf("%d", &num);
   printf("\nEnter the elements of list :");
   for (i = 0; i < num; i++) {
      scanf("%d", &list[i]);
   }
   low = 0;
   high = num - 1;
   printf("\nEnter element to be searched : ");
   scanf("%d", &key);
   position = binsearch(list, key, low, high);
   if (position != -1)
 {
     printf("\nNumber present at %d", (position + 1));
   } else
      printf("\n The number is not present in the list");
   return (0);

}
int binsearch(int a[], int x, int low, int high)
 {
   int mid;
```

```
  if (low > high)
    return -1;
  mid = (low + high) / 2;
  if (x == a[mid])
 {
    return (mid);
  }
 else
 if (x < a[mid])
 {
    binsearch(a, x, low, mid - 1);
  }
 else
 {
    binsearch(a, x, mid + 1, high);
  }
}
```

Output is :-



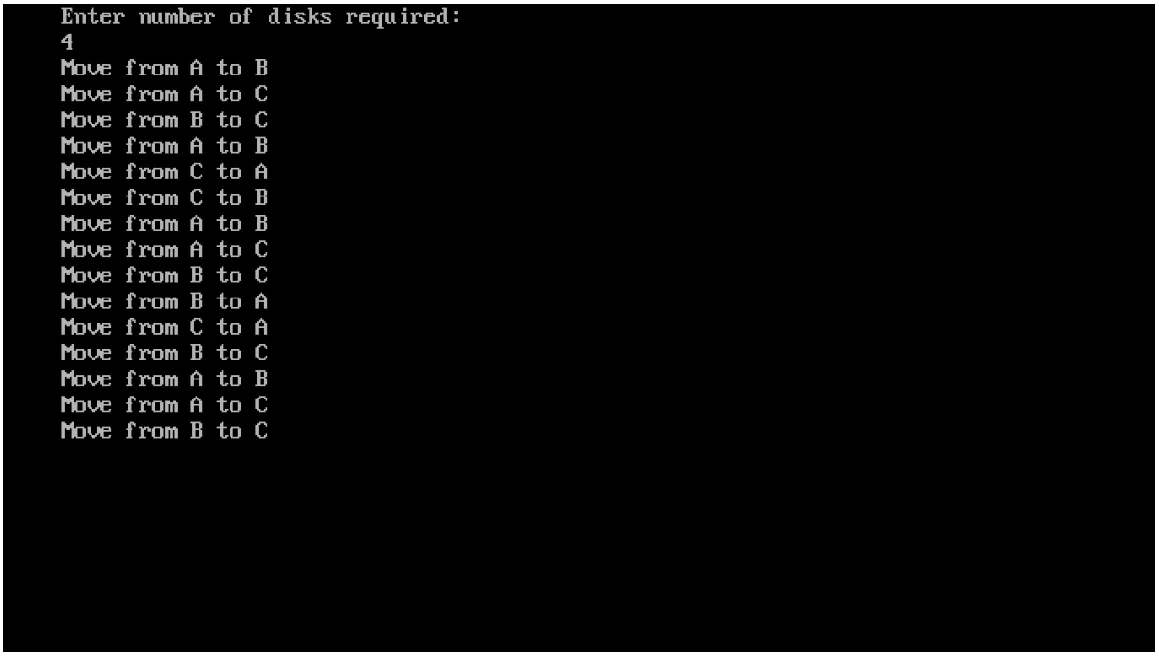
```
Enter the total number of elements
5

Enter the elements of list :
22
73
45
61
12

Enter element to be searched : 45

Number present at 3_
```

Ques 7:- Program for Tower of Hanoi .

Code:-

```
#include<stdio.h>
#include<conio.h>
void TOH(int, char, char, char);
int main ()
{
clrscr();
int n;
printf("Enter number of disks required: \n");
scanf ("%d",  &n);
TOH (n, 'A', 'B', 'C');
getch();
return 0;
}
void TOH (int n, char src, char spare, char dest)
{
if (n==1)
printf("Move from %c to %c \n", src, dest);

else
{
TOH(n-1, src, dest, spare) ;
TOH(1, src, spare, dest);
TOH(n-1, spare, src, dest);
}
}
```

Output is :-



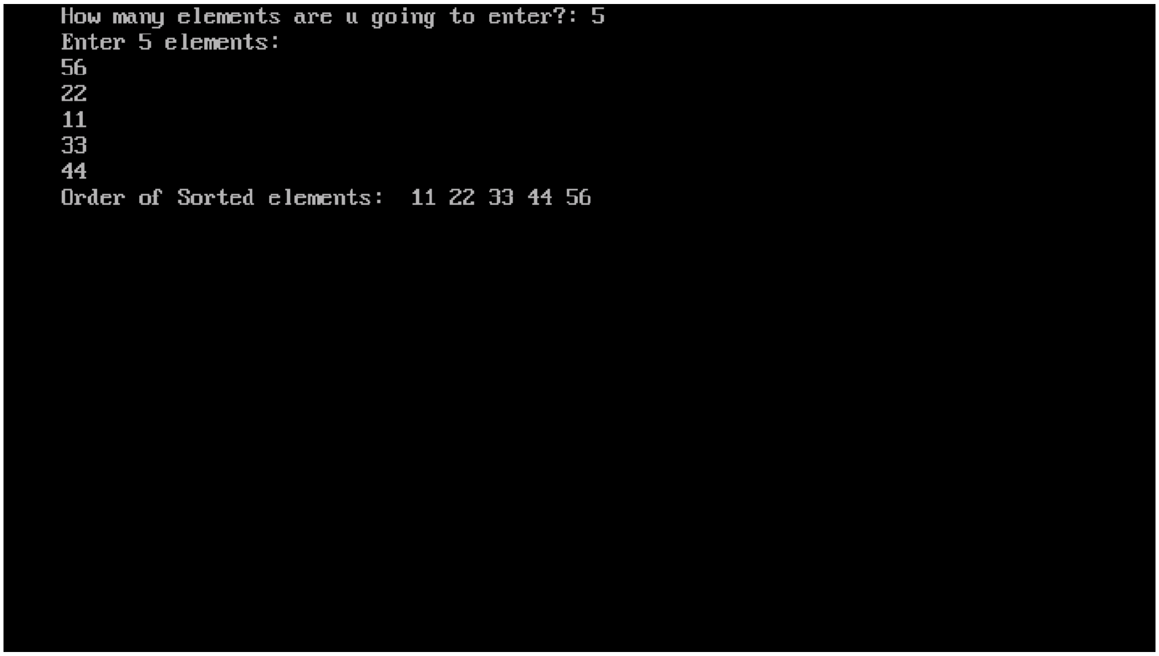
```
Enter number of disks required:
4
Move from A to B
Move from A to C
Move from B to C
Move from A to B
Move from C to A
Move from C to B
Move from A to B
Move from A to C
Move from B to C
Move from B to A
Move from C to A
Move from B to C
Move from A to B
Move from A to C
Move from B to C
```

Ques 8:- Program for Quick sort .

Code :-

```
#include<stdio.h>
#include<conio.h>
void quicksort(int number[25],int first,int last)
{
    int i, j, pivot, temp;
    if(first<last)
  {
    pivot=first;
    i=first;
    j=last;
    while(i<j)
    {
       while(number[i]<=number[pivot] && i<last)
       i++;
       while(number[j]>number[pivot])
       j--;
       if(i<j)
      {
        temp=number[i];
        number[i]=number[j];
        number[j]=temp;
      }
    }
        temp=number[pivot];
        number[pivot]=number[j];
        number[j]=temp;
        quicksort(number,first,j-1);
        quicksort(number,j+1,last);
  }
}

void main()
{
 clrscr();
  int i, count, number[25];
  printf("How many elements are u going to enter?: ");
  scanf("%d",&count);
  printf("Enter %d elements: ", count);
  for(i=0;i<count;i++)
  scanf("%d",&number[i]);
  quicksort(number,0,count-1);
  printf("Order of Sorted elements: ");
  for(i=0;i<count;i++)
  printf(" %d",number[i]);
  getch();
}
```

Output is :-



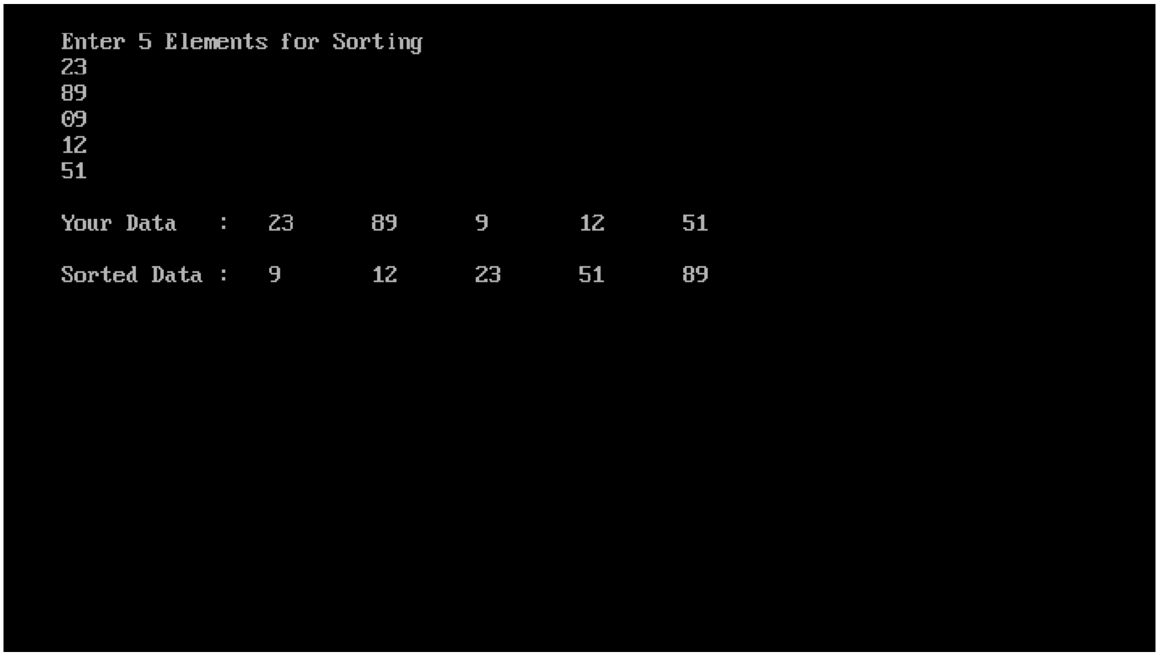
```
How many elements are u going to enter?: 5
Enter 5 elements:
56
22
11
33
44
Order of Sorted elements:  11 22 33 44 56
```

Ques 9:- Program for Merge sort .

Code :-

```
#include<stdio.h>
#include<conio.h>
#define MAX_SIZE 5

void merge_sort(int, int);
void merge_array(int, int, int, int);

int arr_sort[MAX_SIZE];

int main()
 {
  clrscr();
  int i;
  printf("\nEnter %d Elements for Sorting\n", MAX_SIZE);
  for (i = 0; i < MAX_SIZE; i++)
    scanf("%d", &arr_sort[i]);
  printf("\nYour Data   :");
  for (i = 0; i < MAX_SIZE; i++)
 {
    printf("\t%d", arr_sort[i]);
  }

  merge_sort(0, MAX_SIZE - 1);

  printf("\n\nSorted Data :");
  for (i = 0; i < MAX_SIZE; i++)
 {
    printf("\t%d", arr_sort[i]);
  }
  getch();

}

void merge_sort(int i, int j)
 {
  int m;

  if (i < j) {
    m = (i + j) / 2;
    merge_sort(i, m);
    merge_sort(m + 1, j);
    merge_array(i, m, m + 1, j);
  }
}

void merge_array(int a, int b, int c, int d)
 {
  int t[50];
  int i = a, j = c, k = 0;
  while (i <= b && j <= d) {
    if (arr_sort[i] < arr_sort[j])
      t[k++] = arr_sort[i++];
    else
      t[k++] = arr_sort[j++];
  }


  while (i <= b)
    t[k++] = arr_sort[i++];

  while (j <= d)
    t[k++] = arr_sort[j++];

  for (i = a, j = 0; i <= d; i++, j++)
    arr_sort[i] = t[j];
}
```

Output is :-



```
Enter 5 Elements for Sorting
23
89
09
12
51

Your Data   :   23      89      9       12      51

Sorted Data :   9       12      23      51      89
```

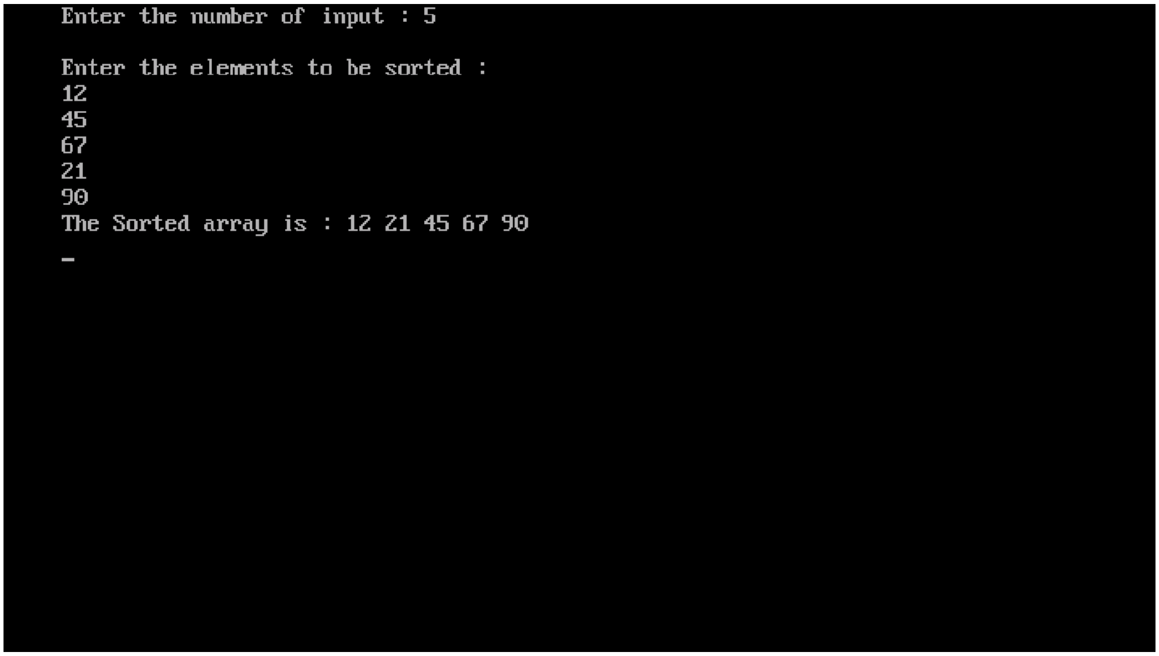
Ques 10:- Program for Counting sort .

Code :-

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

void counting_sort(int A[],

int k, int n)
{
   int i, j;
   int B[15], C[100];
   for (i = 0; i <= k; i++)
      C[i] = 0;
   for (j = 1; j <= n; j++)
      C[A[j]] = C[A[j]] + 1;
   for (i = 1; i <= k; i++)
      C[i] = C[i] + C[i-1];
   for (j = n; j >= 1; j--)
   {
      B[C[A[j]]] = A[j];
      C[A[j]] = C[A[j]] - 1;
   }
   printf("The Sorted array

is : ");
   for (i = 1; i <= n; i++)
      printf("%d ", B[i]);
}

void main()
{
   clrscr();
   int n, k = 0, A[15], i;
   printf("Enter the number

of input : ");
   scanf("%d", &n);
```

```
    printf("\nEnter the

elements to be sorted :\n");
    for (i = 1; i <= n; i++)
    {
            scanf("%d", &A[i]);
            if (A[i] > k) {
                k = A[i];
            }
    }
    counting_sort(A, k, n);
    printf("\n");
    getch();
}
```

Output is :-



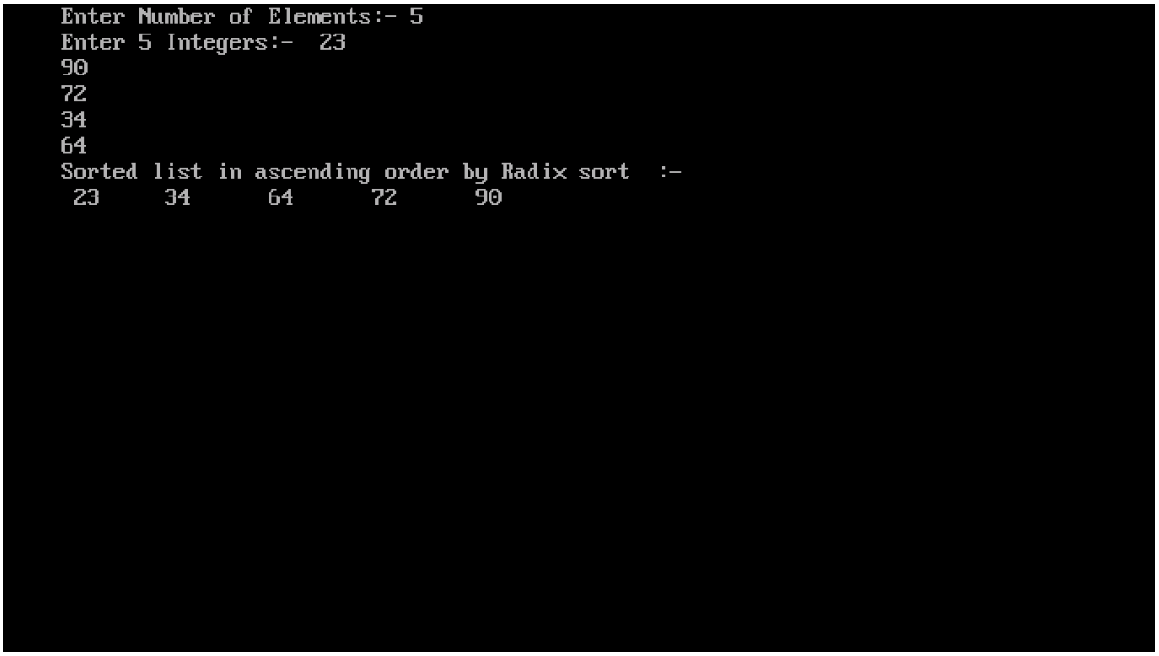
```
Enter the number of input : 5

Enter the elements to be sorted :
12
45
67
21
90
The Sorted array is : 12 21 45 67 90
_
```

Ques 11:- Program for Radix sort .

Code :-

```
#include <stdio.h>
#include <conio.h>
int print(int *a, int n)
 {
 int i;
for (i = 0; i < n; i++)
 printf("%d\t", a[i]);
}

void radix_sort(int *a, int n)
 {
 int i, b[10], m = 0, exp = 1;
 for (i = 0; i < n; i++)
 {
if (a[i] > m)
m = a[i];
}
while (m / exp > 0)
 {
 int box[10] = { 0 };
 for (i = 0; i < n; i++)
box[a[i] / exp % 10]++;
 for (i = 1; i < 10; i++)
 box[i] += box[i - 1];
for (i = n - 1; i >= 0; i--)
b[--box[a[i] / exp % 10]] = a[i];
 for (i = 0; i < n; i++)
a[i] = b[i];
 exp *= 10;
}
}
void main()
 {
 int arr[10];
 int i, num;
 clrscr();
printf("Enter Number of Elements:- ");
 scanf("%d", &num);
printf("Enter %d Integers:-  ", num);
 for (i = 0; i < num; i++)
 scanf("%d", &arr[i]);

radix_sort(&arr[0], num);
printf("Sorted list in ascending order by Radix sort  :- \n ");
 print(&arr[0], num);
getch();
}
```

Output is :-



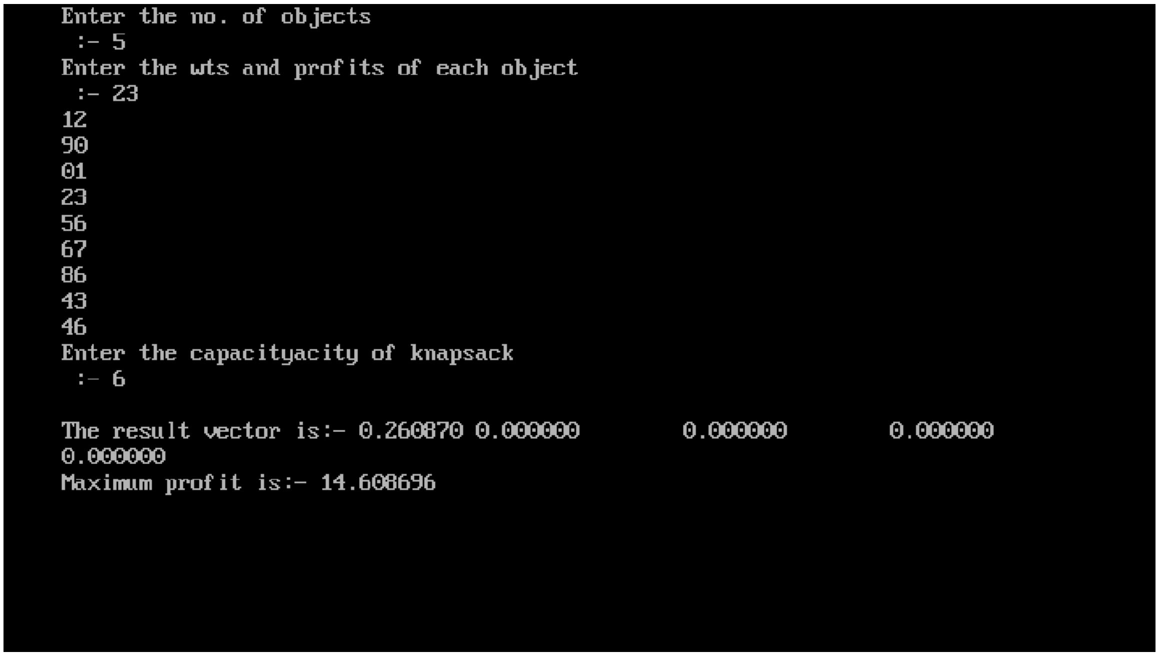
```
Enter Number of Elements:- 5
Enter 5 Integers:-  23
90
72
34
64
Sorted list in ascending order by Radix sort  :-
 23     34      64      72      90
```

Ques 12:- Program for fractional Knapsack problem using greedy method .

Code :-

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

void knapsack(int n, float weight[], float profit[], float capacity)
{
   float x[20], tp = 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;
tp = tp + profit[i]; u = u - weight[i];
}
}
if (i < n)
x[i] = u / weight[i];
tp = tp + (x[i] * profit[i]);
printf("\nThe result vector is:- ");
for (i = 0; i < n; i++)
printf("%f\t", x[i]);
printf("\nMaximum profit is:- %f", tp);
}
void main()
{
clrscr();
float weight[20], profit[20], capacity; int num, i, j;
float ratio[20], temp;
printf("Enter the no. of objects \n :- ");
scanf("%d", &num);
printf("Enter the wts and profits of each object \n :- ");
for (i = 0; i < num; i++)
{
scanf("%f %f", &weight[i], &profit[i]);
}
printf("Enter the capacityacity of knapsack \n :- ");
scanf("%f", &capacity);
for (i = 0; i < num; i++)
{
   ratio[i] = profit[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 = profit[j];
profit[j] = profit[i];
profit[i] = temp;
}
}
}
knapsack(num, weight, profit, capacity);
getch();
}
```

Output is :-



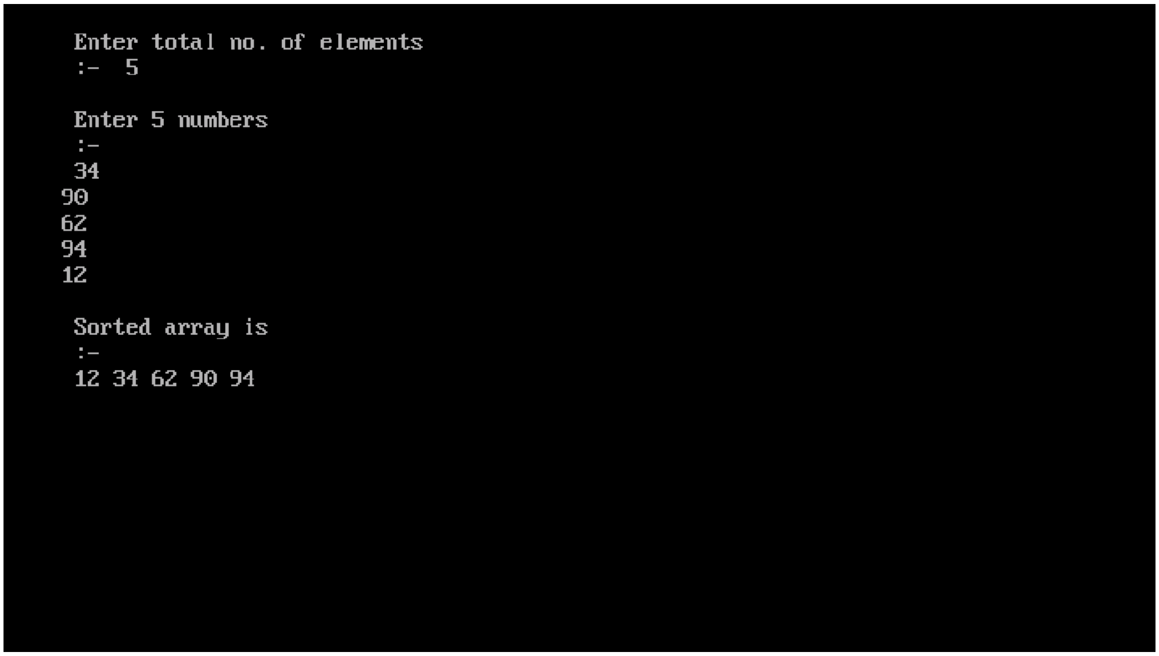
```
Enter the no. of objects
 :- 5
Enter the wts and profits of each object
 :- 23
12
90
01
23
56
67
86
43
46
Enter the capacityacity of knapsack
 :- 6

The result vector is:- 0.260870 0.000000        0.000000        0.000000
0.000000
Maximum profit is:- 14.608696
```

Ques 13:- Program for Shell sort .

Code :-

```
#include <stdio.h>
#include <conio.h>
void shellsort(int arr[], int num)
{
    int i, j, k, tmp;
    for (i = num / 2; i > 0; i = i / 2)
    {
        for (j = i; j < num; j++)
        {
            for(k = j - i; k >= 0; k = k - i)
            {
                if (arr[k+i] >= arr[k])
                    break;
                else
                {
                    tmp = arr[k];
                    arr[k] = arr[k+i];
                    arr[k+i] = tmp;
                }
            }
        }
    }
}
void main()
{
    clrscr();
    int arr[50];
    int n,  num;
    printf(" \n Enter total no. of elements \n :-  ");
    scanf("%d", &num);
    printf(" \n Enter %d numbers \n :- \n ", num);

    for (n =  0 ; n < num; n++)
    {
            scanf("%d", &arr[n]);
    }
    shellsort(arr, num);
    printf(" \n Sorted array is \n :- \n ");
    for (n = 0; n < num; n++)
        printf("%d ", arr[n]);
    getch();
}
```

Output is :-



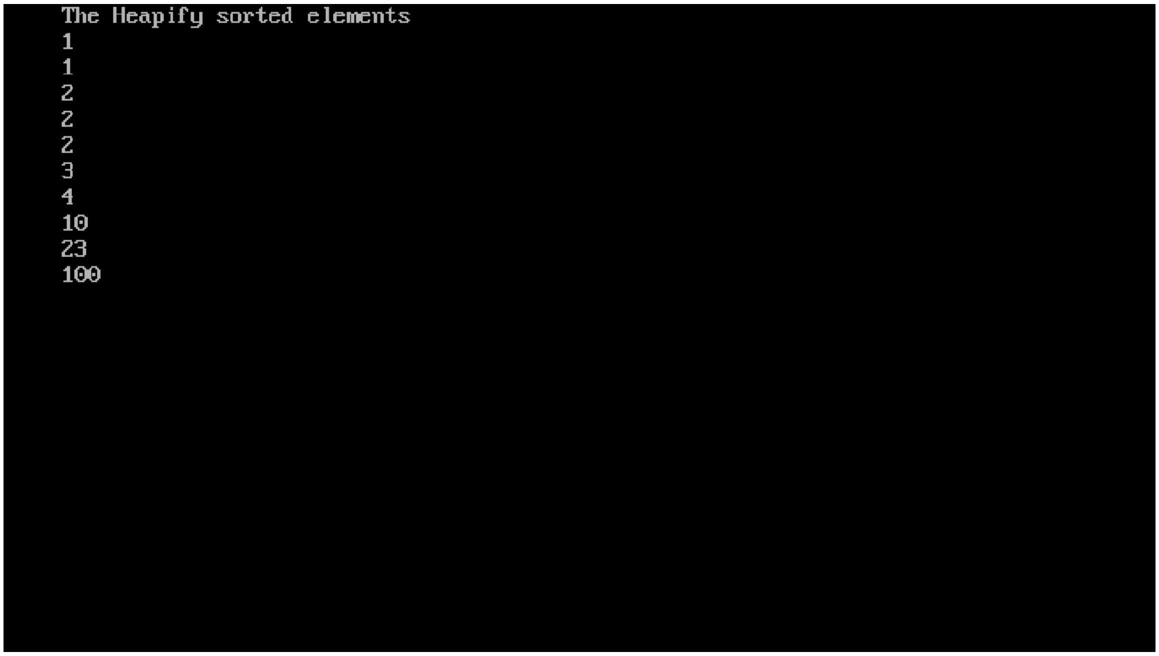
```
 Enter total no. of elements
 :-  5

 Enter 5 numbers
 :-
 34
90
62
94
12

 Sorted array is
 :-
 12 34 62 90 94
```

Ques 14 :- Program for Heap sort .

Code :-

```
#include<stdio.h>
#include<conio.h>
int temp;

void heapify(int arr[], int size, int i)
{
int largest = i;
int left = 2*i + 1;
int right = 2*i + 2;

if (left < size && arr[left] >arr[largest])
largest = left;

if (right < size && arr[right] > arr[largest])
largest = right;

if (largest != i)
{
temp = arr[i];
    arr[i]= arr[largest];
    arr[largest] = temp;
heapify(arr, size, largest);
}
}

void heapSort(int arr[], int size)
{
int i;
for (i = size / 2 - 1; i >= 0; i--)
heapify(arr, size, i);
for (i=size-1; i>=0; i--)
{
temp = arr[0];
    arr[0]= arr[i];
    arr[i] = temp;
heapify(arr, i, 0);
}
}

void main()
{
clrscr();
int arr[] = {1, 10, 2, 3, 4, 1, 2, 100,23, 2};
int i;
int size = sizeof(arr)/sizeof(arr[0]);

heapSort(arr, size);

printf("The Heapify sorted elements\n");
for (i=0; i<size; ++i)
printf("%d\n",arr[i]);
}
```

Output is :-



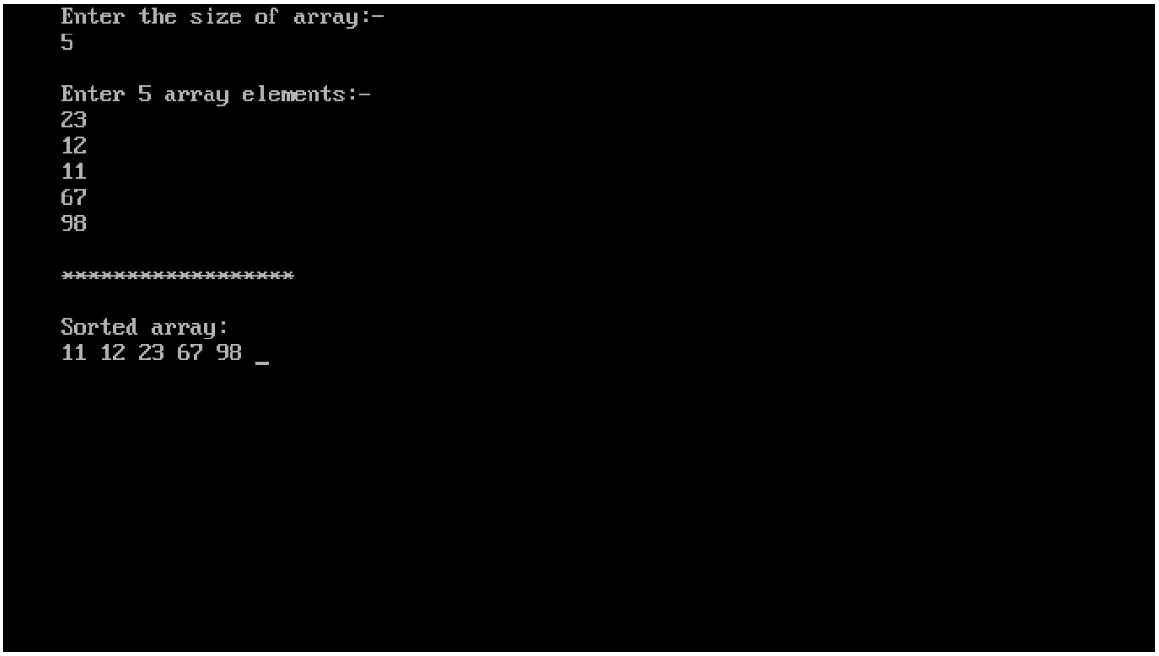
```
The Heapify sorted elements
1
1
2
2
2
3
4
10
23
100
```

Ques 15:- Program for Tree sort .

Code :-

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
struct node{
        int info;
        struct node *lp;
        struct node *rp;
};
void inorder(int arr[], struct node* root)
{
if(root!=NULL)
{
static int i = 0;
inorder(arr,root->lp);
arr[i++]=root->info;
inorder(arr,root->rp);
}
}
void main()
{
clrscr();
int arr[10],n;
printf("Enter the size of array:- \n");
scanf("%d",&n);
printf("\nEnter %d array elements:- \n",n);
for(int i = 0; i<n; i++)
scanf("%d",&arr[i]);
struct node *head = (struct node *)malloc(sizeof(struct node));
struct node *ptr = (struct node *)malloc(sizeof(struct node));
ptr->info = arr[0];
ptr->lp = NULL;
ptr->rp = NULL;
head = ptr;
for(i = 1; i<n; i++)
{
        ptr = head;
        struct node *next = (struct node *)malloc(sizeof(struct node));
        next->info = arr[i];
        next->lp = NULL;
        next->rp = NULL;
        int flag;
        do
{
        flag = 0;
                if((next->info)<(ptr->info))
                {
                        if(ptr->lp==NULL)
                        {
                        ptr->lp = next;
                        flag = 1;
                        }
                        else
{
                        ptr = ptr->lp;
                        }
                }
                else
                {
                        if(ptr->rp==NULL)
                        {
                        ptr->rp = next;
                        flag = 1;
                        }
                        else
{
                        ptr = ptr->rp;
                        }
                }
}
while(flag==0);
}
inorder(arr,head);
printf("\n******************\n");
printf("\nSorted array:\n");
for(i = 0; i<n; i++)
printf("%d ",arr[i]);
getch();
}
```

Output is :-



```
Enter the size of array:-
5

Enter 5 array elements:-
23
12
11
67
98

******************

Sorted array:
11 12 23 67 98 _
```

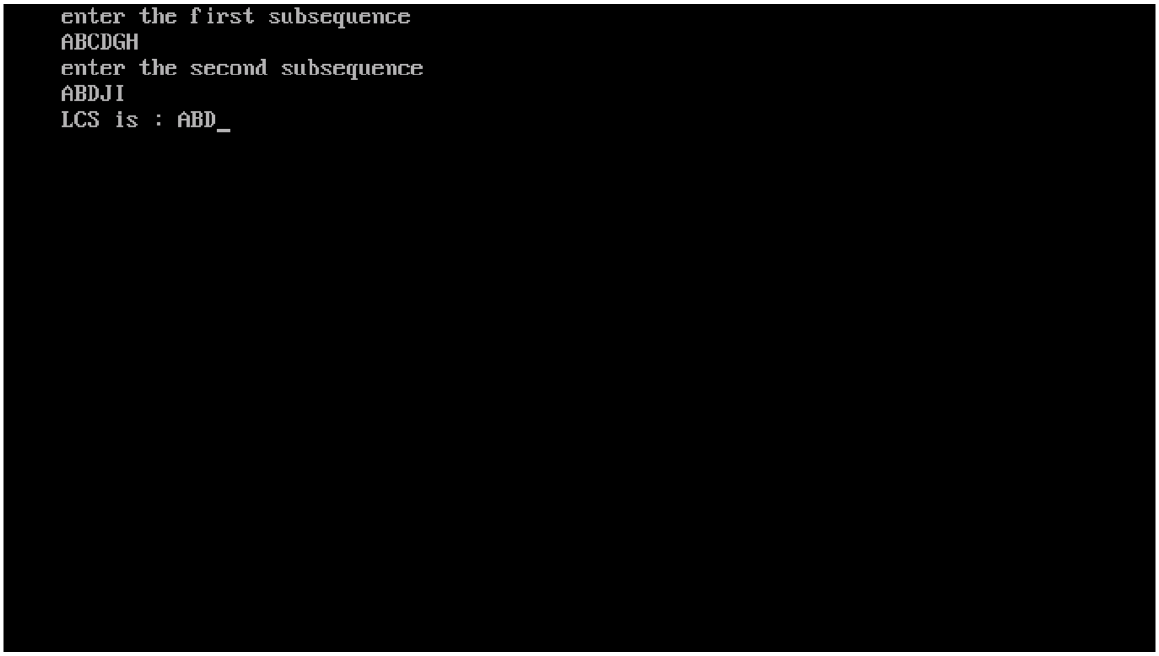
Ques 16:- Program for Longest common subsequence .

Code :-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
int LCS();
int display(int, int);
int i, j, p, q;
char G[20], H[20], b[20][20], c[20][20];
void main()
{
            clrscr();
        printf("enter the first subsequence\n");
        gets(G);
        printf("enter the second subsequence\n");
        gets(H);
        printf("LCS is : ");
        LCS();
        display(p, q);
}
int LCS()
{
       p = strlen(G);
        q = strlen(H);
        for(i=0;i<=p;i++)
        {
                c[i][0] = 0;
        }
        for(i=0;i<=q;i++)
        {
                 c[0][i] = 0;
        }
        for(i=1;i<=p;i++)
        {
                for(j=1;j<=q;j++)
                {
```

```
            if(G[i-1] == H[j-1])
            {
                c[i][j] = c[i-1][j-1] + 1;

                b[i][j] = 'c';
            }
            else if(c[i-1][j] >= c[i][j-1])
            {
                c[i][j] = c[i-1][j];
                b[i][j] = 'u';
            }
            else
            {
                c[i][j] = c[i][j-1];
                b[i][j] = 'l';
            }
        }
    }
    return 0;
}
int display(int i, int j)
{
    if(i==0 || j==0)
    {
            return 0;
    }

    if(b[i][j] == 'c')
    {
        display(i-1, j-1);
        printf("%c",G[i-1]);
    }
    else if(b[i][j] == 'u')
    {
        display(i-1, j);
    }
    else
    {
        display(i, j-1);
    }
    return 0;
}
```

Output is:-



```
enter the first subsequence
ABCDGH
enter the second subsequence
ABDJI
LCS is : ABD_
```

Ques 17:- Program for Matrix chain multiplication .

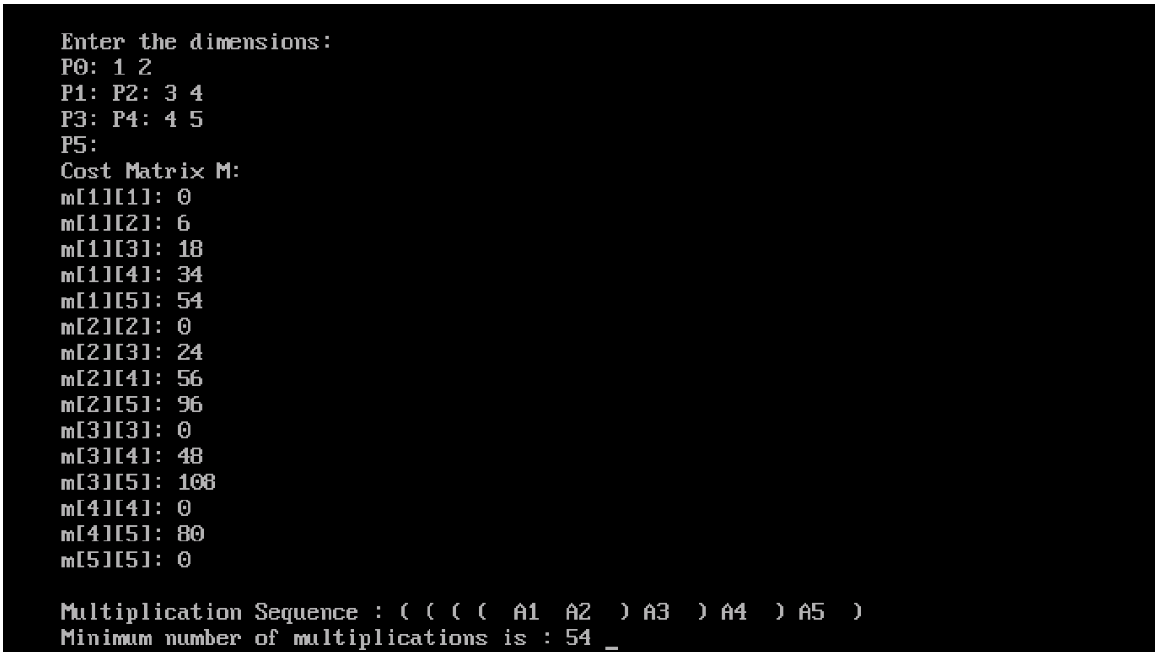
Code :-

```
#include <stdio.h>
#include <conio.h>
#include <limits.h>
#define INFY 999999999
long int m[20][20];
int s[20][20];
int p[20],i,j,n;
void print_optimal(int i,int j)
{
if (i == j)
printf(" A%d ",i);
 else
{
printf("( ");
print_optimal(i, s[i][j]);
 print_optimal(s[i][j] + 1, j);
 printf(" )");
}
}
void matmultiply(void)
{
long int q;
 int k;
for(i=n;i>0;i--)
{
for(j=i;j<=n;j++)
{
if(i==j) m[i][j]=0;
 else
{
for(k=i;k<j;k++)
{
q=m[i][k]+m[k+1][j]+p[i-1]*p[k]*p[j];
 if(q<m[i][j])
{
```

```
m[i][j]=q;
s[i][j]=k;
}
}
}
}
}
}
int MatrixChainOrder(int p[], int i, int j)
{
if(i == j)
 return 0;
int k;
int min = INT_MAX;
 int count;
for (k = i; k <j; k++)
{
count = MatrixChainOrder(p, i, k) + MatrixChainOrder(p, k+1, j) + p[i-1]*p[k]*p[j];
if (count < min) min = count;
}
return min;
}
void main()
{
clrscr();
int k;
printf("Enter the no. of elements: ");
 scanf("%d",&n);
for(i=1;i<=n;i++)
 for(j=i+1;j<=n;j++)
{
m[i][i]=0;
m[i][j]=INFY;
s[i][j]=0;
}
printf("\nEnter the dimensions: \n");
 for(k=0;k<=n;k++)
{
```

```
printf("P%d: ",k);
scanf("%d",&p[k]);
}
matmultiply();
printf("\nCost Matrix M:\n");
 for(i=1;i<=n;i++)
 for(j=i;j<=n;j++)
printf("m[%d][%d]: %ld\n",i,j,m[i][j]);
 i=1,j=n;
printf("\nMultiplication Sequence : ");
 print_optimal(i,j);
printf("\nMinimum number of multiplications is : %d ", MatrixChainOrder(p, 1, n));
}
```

Output is:-



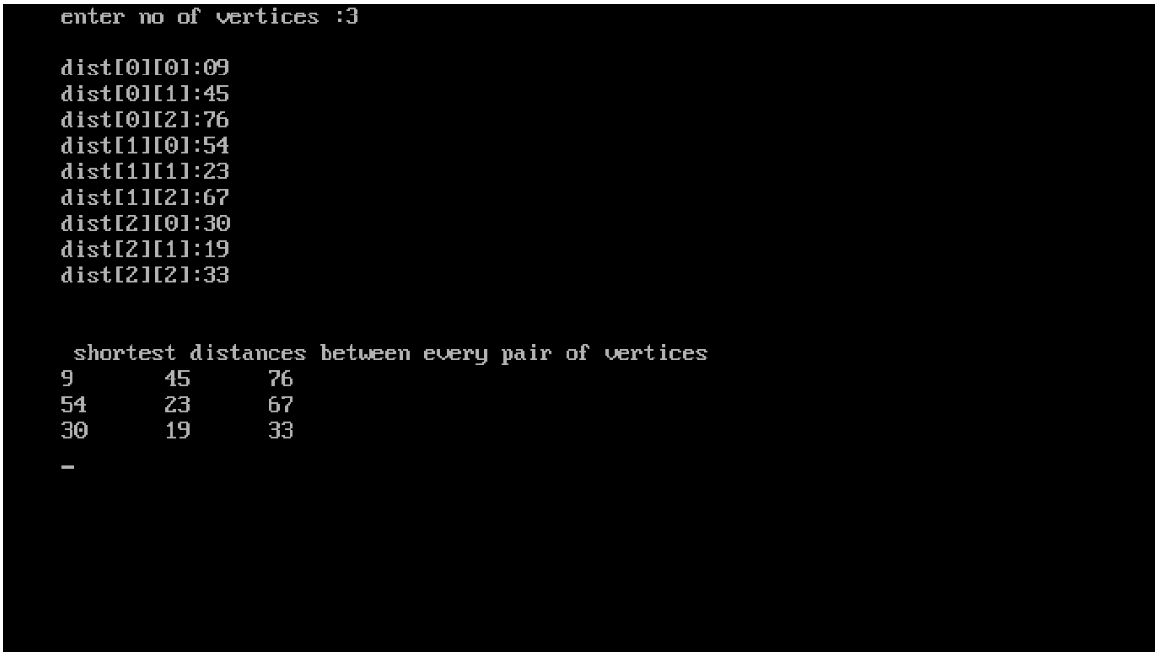
```
Enter the dimensions:
P0: 1 2
P1: P2: 3 4
P3: P4: 4 5
P5:
Cost Matrix M:
m[1][1]: 0
m[1][2]: 6
m[1][3]: 18
m[1][4]: 34
m[1][5]: 54
m[2][2]: 0
m[2][3]: 24
m[2][4]: 56
m[2][5]: 96
m[3][3]: 0
m[3][4]: 48
m[3][5]: 108
m[4][4]: 0
m[4][5]: 80
m[5][5]: 0

Multiplication Sequence : ( ( ( (  A1  A2  ) A3  ) A4  ) A5  )
Minimum number of multiplications is : 54 _
```

Ques 18:- Program for Floyd Warshall all pair shortest path .

Code :-

```
#include<stdio.h>
#include<conio.h>
int i, j, k,n,dist[10][10];
void floydWarshell ()
{
 for (k = 0; k < n; k++)
  for (i = 0; i < n; i++)
   for (j = 0; j < n; j++)
    if (dist[i][k] + dist[k][j] < dist[i][j])
     dist[i][j] = dist[i][k] + dist[k][j];
}
int main()
{
  clrscr();
  int i,j;
  printf("enter no of vertices :");
  scanf("%d",&n);
  printf("\n");
  for(i=0;i<n;i++)
  for(j=0;j<n;j++)
   {
    printf("dist[%d][%d]:",i,j);
    scanf("%d",&dist[i][j]);
   }
 floydWarshell();
 printf (" \n\n shortest distances between every pair of vertices \n");
 for(i = 0; i < n; i++)
 {
  for(j = 0; j < n; j++)
   printf ("%d\t", dist[i][j]);
  printf("\n");
 }
 return 0;
}
```

Output is :-



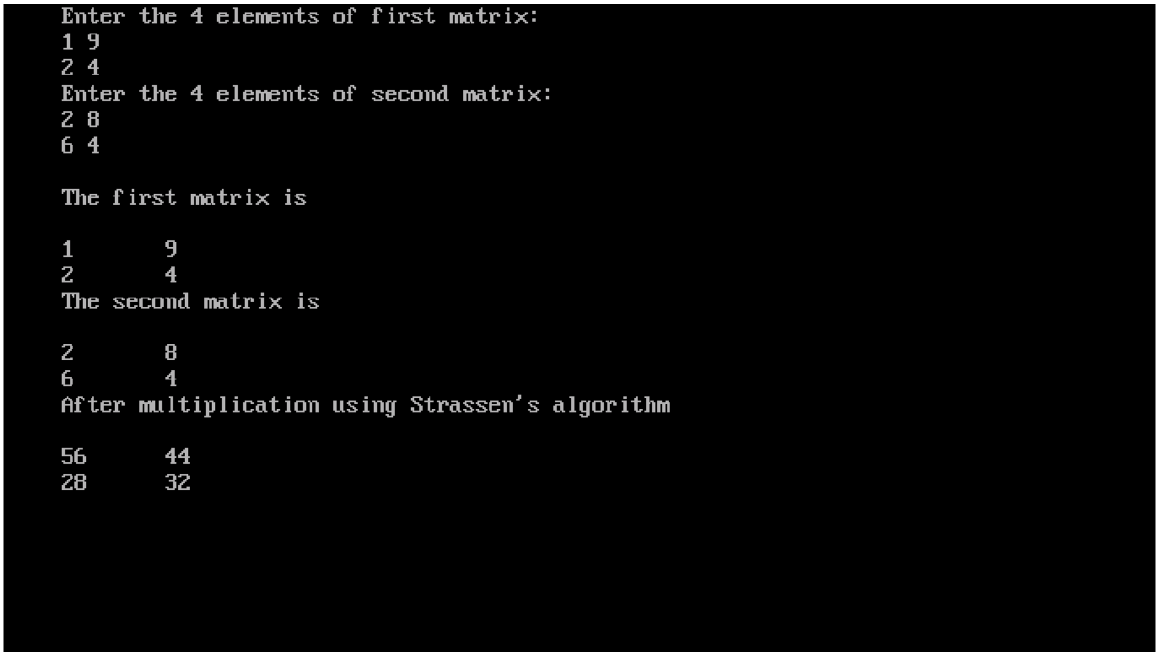
```
enter no of vertices :3

dist[0][0]:09
dist[0][1]:45
dist[0][2]:76
dist[1][0]:54
dist[1][1]:23
dist[1][2]:67
dist[2][0]:30
dist[2][1]:19
dist[2][2]:33


 shortest distances between every pair of vertices
9       45      76
54      23      67
30      19      33
_
```

Ques 19:- Program for strassean matrix multiplication .

Code :-

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
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\n");
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\n");
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 using Strassen's algorithm \n");
for(i = 0; i < 2 ; i++)
{
printf("\n");
for(j = 0;j < 2; j++)
printf("%d\t", c[i][j]);
}
getch();
}
```

Output is :-



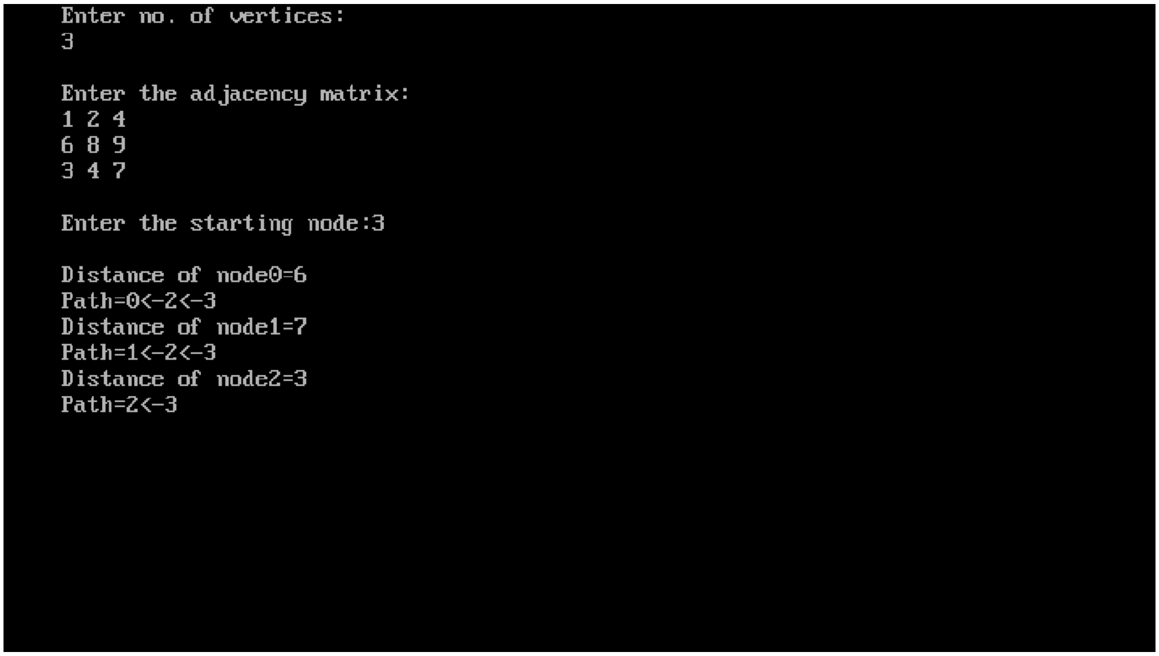
```
Enter the 4 elements of first matrix:
1 9
2 4
Enter the 4 elements of second matrix:
2 8
6 4

The first matrix is

1       9
2       4
The second matrix is

2       8
6       4
After multiplication using Strassen's algorithm

56      44
28      32
```

Ques 20:- Program for Dijkstra algorithm .

Code :-

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

void dijkstra(int G[MAX][MAX],int n,int startnode);
 int main()
{
clrscr();
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;
}
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 is :-



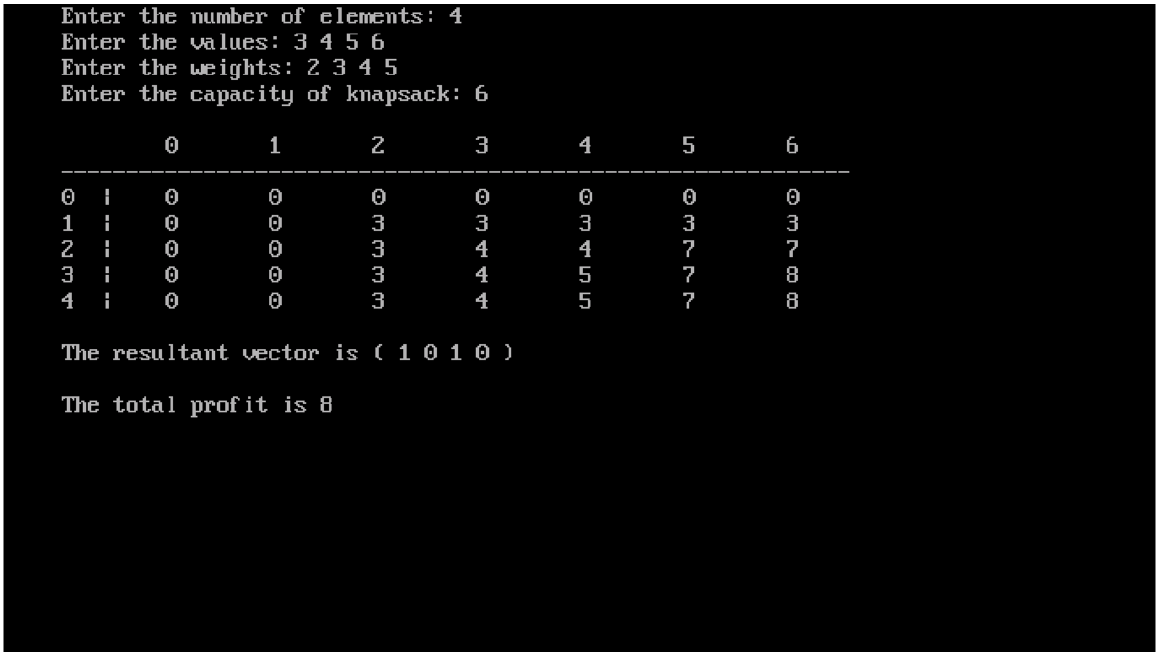
```
Enter no. of vertices:
3

Enter the adjacency matrix:
1 2 4
6 8 9
3 4 7

Enter the starting node:3

Distance of node0=6
Path=0<-2<-3
Distance of node1=7
Path=1<-2<-3
Distance of node2=3
Path=2<-3
```

Ques 21:- Program for0/1 knapsack problem by dynamic programming.

Code :-

```
#include<stdio.h>
#include<conio.h>
#define MAX 100
int main()
{
 int n,flag[MAX]={0},v[MAX],w[MAX],m[MAX][MAX],W,i,j,k;
 clrscr();
 printf("Enter the number of elements: ");
 scanf("%d",&n);
 printf("Enter the values: ");
 for(i=1;i<=n;i++)
  scanf("%d",&v[i]);
 printf("Enter the weights: ");
 for(i=1;i<=n;i++)
  scanf("%d",&w[i]);
 printf("Enter the capacity of knapsack: ");
 scanf("%d",&W);
 for(j=0;j<=W;j++)
   m[0][j]=0;
 for(i=1;i<=n;i++)
 {
  for(j=0;j<=W;j++)
  {
   if(w[i]<=j)
   {
    if( m[i-1][j] > (m[i-1][j-w[i]]+v[i]) )
     m[i][j]=m[i-1][j];
    else
     m[i][j]=m[i-1][j-w[i]]+v[i];
   }
   else
    m[i][j]=m[i-1][j];
  }
 }
 i=n;
 k=W;
 while(i>0 && k>0)
 {
  if(m[i][k]!=m[i-1][k])
  {
   flag[i]=1;
   k=k-w[i];
   i=i-1;
  }
  else
   i--;

 }
 printf("\n\t");
 for(i=0;i<=W;i++)
  printf("%d\t",i);
 printf("\n");
 for(i=0;i<=10*W;i++)
  printf("-");
 printf("\n");
 for(i=0;i<=n;i++)
 {
  printf("%d  |\t", i); //to print the vertical line
  for(j=0;j<=W;j++)
   printf("%d\t",m[i][j]);
  printf("\n");
 }
 printf("\nThe resultant vector is ");
 printf("( ");
 for(i=1;i<=n;i++)
  printf("%d ",flag[i]);
 printf(")");
 printf("\n\nThe total profit is %d",m[n][W]);
 printf("\n");
 getch();
 return 0;
}
```

Output is :-



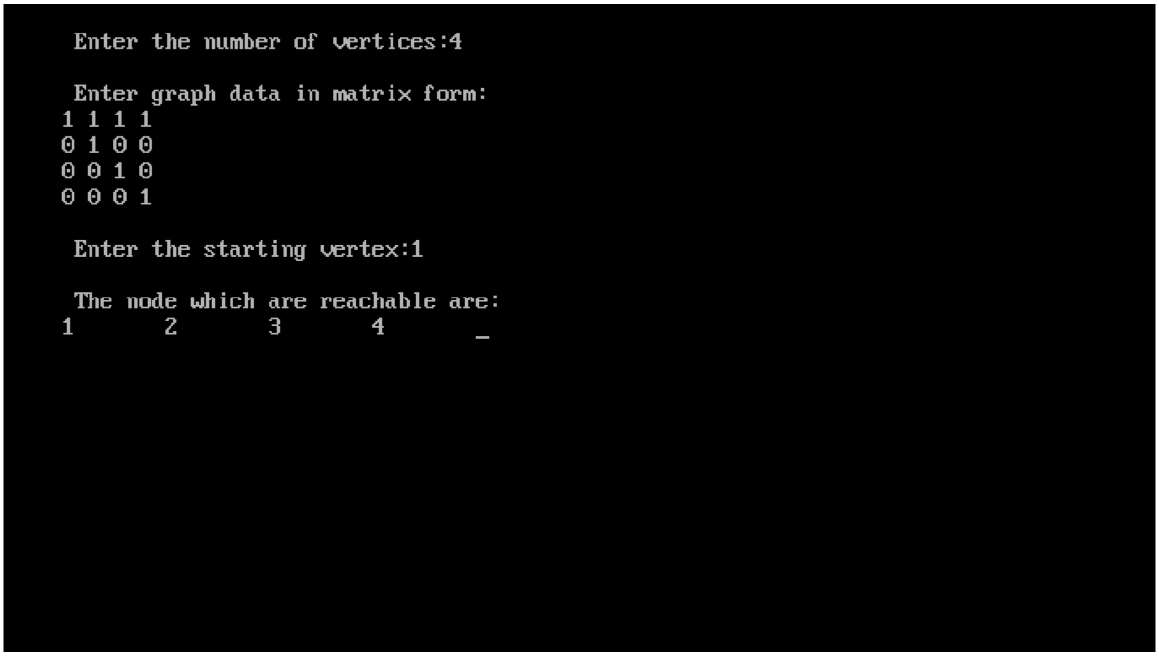
```
Enter the number of elements: 4
Enter the values: 3 4 5 6
Enter the weights: 2 3 4 5
Enter the capacity of knapsack: 6

        0       1       2       3       4       5       6
-------------------------------------------------------------
0  ¦    0       0       0       0       0       0       0
1  ¦    0       0       3       3       3       3       3
2  ¦    0       0       3       4       4       7       7
3  ¦    0       0       3       4       5       7       8
4  ¦    0       0       3       4       5       7       8

The resultant vector is ( 1 0 1 0 )

The total profit is 8
```

Ques 22:- Program for Graph Traversal using BFS .

Code :-

```
#include<stdio.h>
#include<conio.h>
int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;
void bfs(int v)
 {
        for(i = 1; i <= n; i++)
                if(a[v][i] && !visited[i])
                        q[++r] = i;
        if(f <= r)
 {
                visited[q[f]] = 1;
                bfs(q[f++]);
        }
}
void main()
 {
        clrscr();
        int v;
        printf("\n Enter the number of vertices:");
        scanf("%d", &n);
        for(i=1; i <= n; i++)
 {
                q[i] = 0;
                visited[i] = 0;
        }
        printf("\n Enter graph data in matrix form:\n");
        for(i=1; i<=n; i++)
 {
                for(j=1;j<=n;j++)
 {
                scanf("%d", &a[i][j]);
                }
        }
        printf("\n Enter the starting vertex:");
        scanf("%d", &v);
        bfs(v);
        printf("\n The node which are reachable are:\n");
        for(i=1; i <= n; i++)
  {
                if(visited[i])
                        printf("%d\t", i);
                else
 {
                printf("\n Bfs is not possible. Not all nodesare not reachable");
                        break;
                }
 }
 }
```

Output is :-



```
 Enter the number of vertices:4

 Enter graph data in matrix form:
1 1 1 1
0 1 0 0
0 0 1 0
0 0 0 1

 Enter the starting vertex:1

 The node which are reachable are:
1       2       3       4       _
```

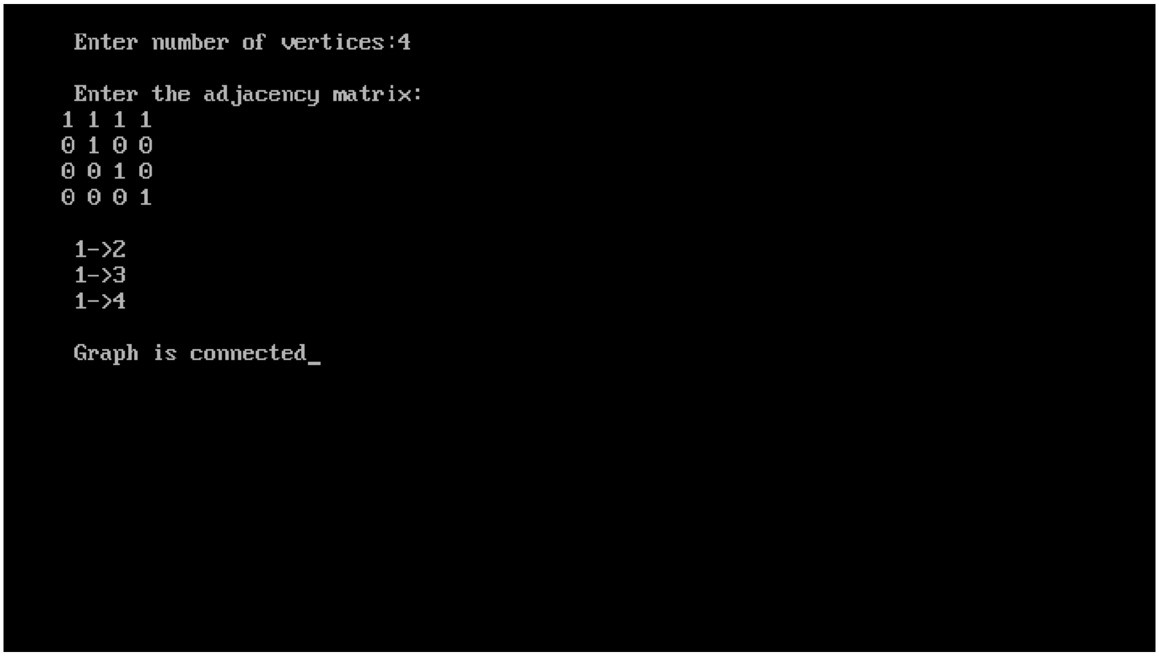
Ques 23:- Program for Graph Traversal using DFS .

Code :-

```
#include<stdio.h>
#include<conio.h>
int a[20][20],reach[20],n;
void dfs(int v)
 {
            int i;
            reach[v]=1;
            for (i=1;i<=n;i++)
              if(a[v][i] && !reach[i])
{
                        printf("\n %d->%d",v,i);
                        dfs(i);
            }
}

void main()
 {
            int i,j,count=0;
            clrscr();
            printf("\n Enter number of vertices:");
            scanf("%d",&n);
            for (i=1;i<=n;i++)
 {
                        reach[i]=0;
                        for (j=1;j<=n;j++)
                           a[i][j]=0;
            }
            printf("\n Enter the adjacency matrix:\n");
            for (i=1;i<=n;i++)
              for (j=1;j<=n;j++)
               scanf("%d",&a[i][j]);
            dfs(1);
            printf("\n");
            for (i=1;i<=n;i++)
 {
```

```
                    if(reach[i])
                      count++;
            }
            if(count==n)
              printf("\n Graph is connected");
 else
              printf("\n Graph is not connected");
            getch();
}
```

Output is :-



```
 Enter number of vertices:4

 Enter the adjacency matrix:
1 1 1 1
0 1 0 0
0 0 1 0
0 0 0 1

 1->2
 1->3
 1->4

 Graph is connected_
```

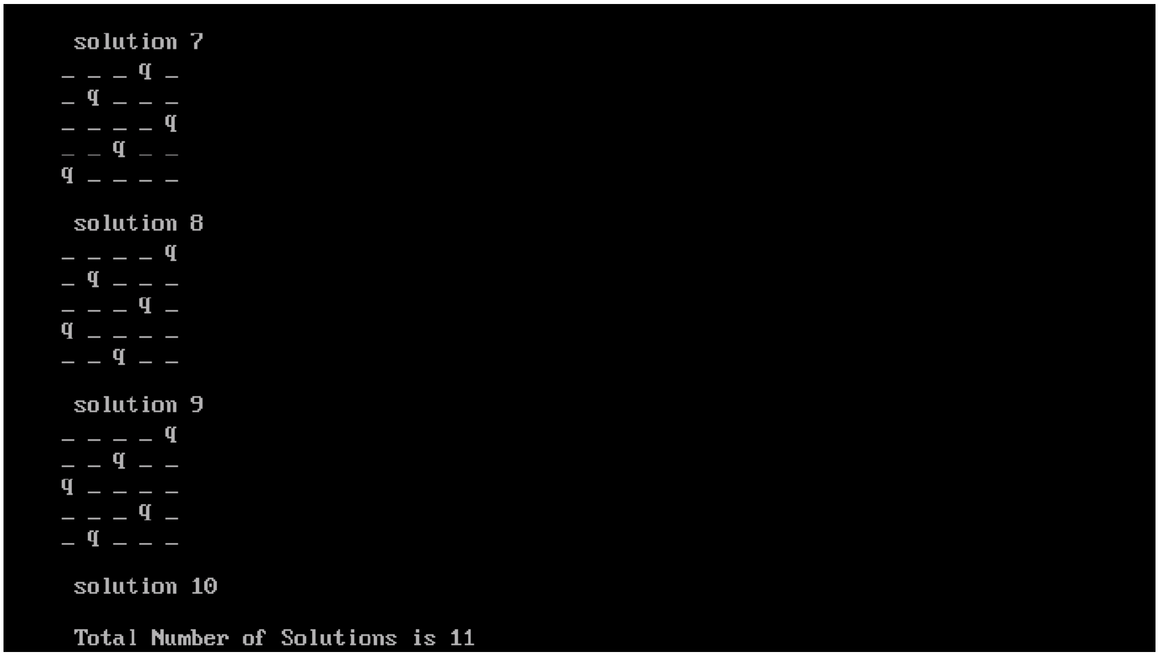
Ques 24:- Program for N Queen’s Problem .

Code :-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int t[5] = {-1};
int sol = 1;
 void printsol()
{
int i,j;
char crossboard[5][5];
 for(i=0;i<5;i++)
{
for(j=0;j<5;j++)
{
crossboard[i][j]='_';
}
}
for(i=0;i<5;i++)
{
crossboard[i][t[i]]='q';
}
for(i=0;i<5;i++)
{
for(j=0;j<5;j++)
{
printf("%c ",crossboard[i][j]);
}
printf("\n");
}
}
int empty(int i)
{
int j=0;
while((t[i]!=t[j])&&(abs(t[i]-t[j])!=(i-j))&&j<5)j++;
 return i==j?1:0;
}
```

```
void queens(int i)
{
for(t[i] = 0;t[i]<5;t[i]++)
{
if(empty(i))
{
if(i==4)
{
 printsol();
printf("\n solution %d\n",sol++);
}
else queens(i+1);
}
}
}
int main()
{
clrscr();
queens(0);
printf("\n Total Number of Solutions is %d",sol);
 return 0;
}
```

Output is :-



```
  solution 7
_ _ _ q _
_ q _ _ _
_ _ _ _ q
_ _ q _ _
q _ _ _ _

  solution 8
_ _ _ _ q
_ q _ _ _
_ _ _ q _
q _ _ _ _
_ _ q _ _

  solution 9
_ _ _ _ q
_ _ q _ _
q _ _ _ _
_ _ _ q _
_ q _ _ _

  solution 10

  Total Number of Solutions is 11
```

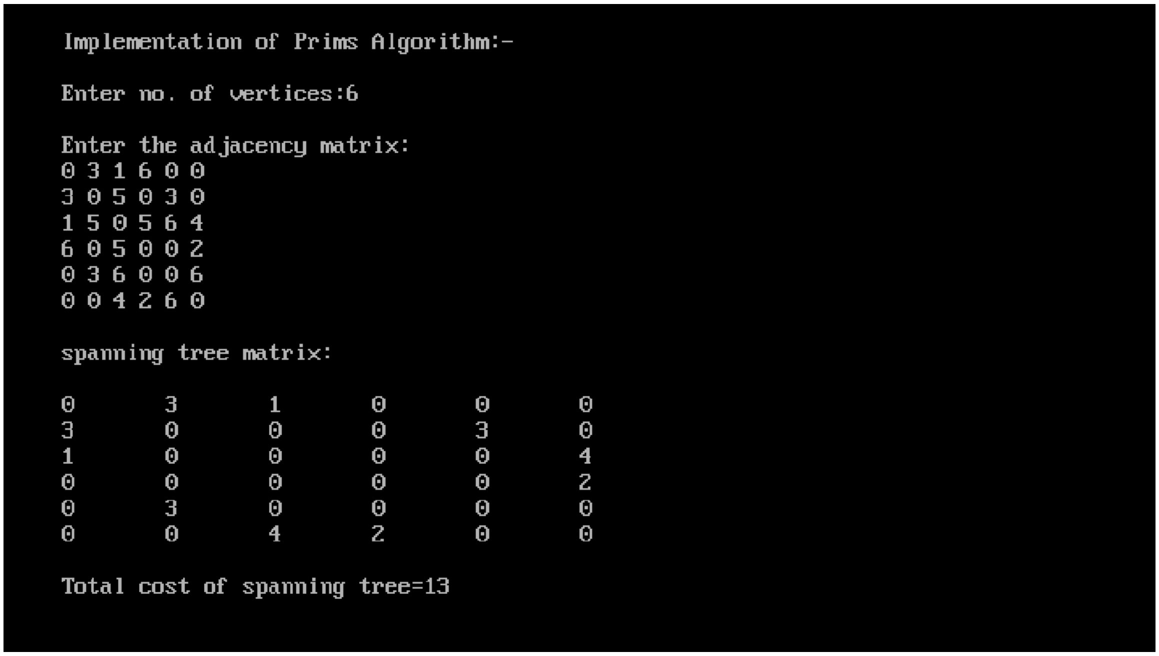
Ques 25:- Program for MST using prim’s algorithm .

Code :-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define infinity 9999
#define MAX 20

int G[MAX][MAX],spanning[MAX][MAX],n;

int prims();

int main()
{
   clrscr();
   int i,j,total_cost;
   printf("\nImplementation of Prims Algorithm:-\n");
   printf("\nEnter 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]);

   total_cost=prims();
   printf("\nspanning tree matrix:\n");

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

   printf("\n\nTotal cost of spanning tree=%d",total_cost);
```

```
    return 0;
}

int prims()
{
    int cost[MAX][MAX];
    int u,v,min_distance,distance[MAX],from[MAX];
    int visited[MAX],no_of_edges,i,min_cost,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];
                spanning[i][j]=0;
        }

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

    for(i=1;i<n;i++)
    {
        distance[i]=cost[0][i];
        from[i]=0;
        visited[i]=0;
    }

    min_cost=0;
    no_of_edges=n-1;

    while(no_of_edges>0)
    {
                min_distance=infinity;
        for(i=1;i<n;i++)
            if(visited[i]==0&&distance[i]<min_distance)
            {
                v=i;
                min_distance=distance[i];
            }

        u=from[v];

        spanning[u][v]=distance[v];
        spanning[v][u]=distance[v];
        no_of_edges--;
        visited[v]=1;

                for(i=1;i<n;i++)
            if(visited[i]==0&&cost[i][v]<distance[i])
            {
                distance[i]=cost[i][v];
                from[i]=v;
            }

        min_cost=min_cost+cost[u][v];
    }

    return(min_cost);
}
```

Output is :-



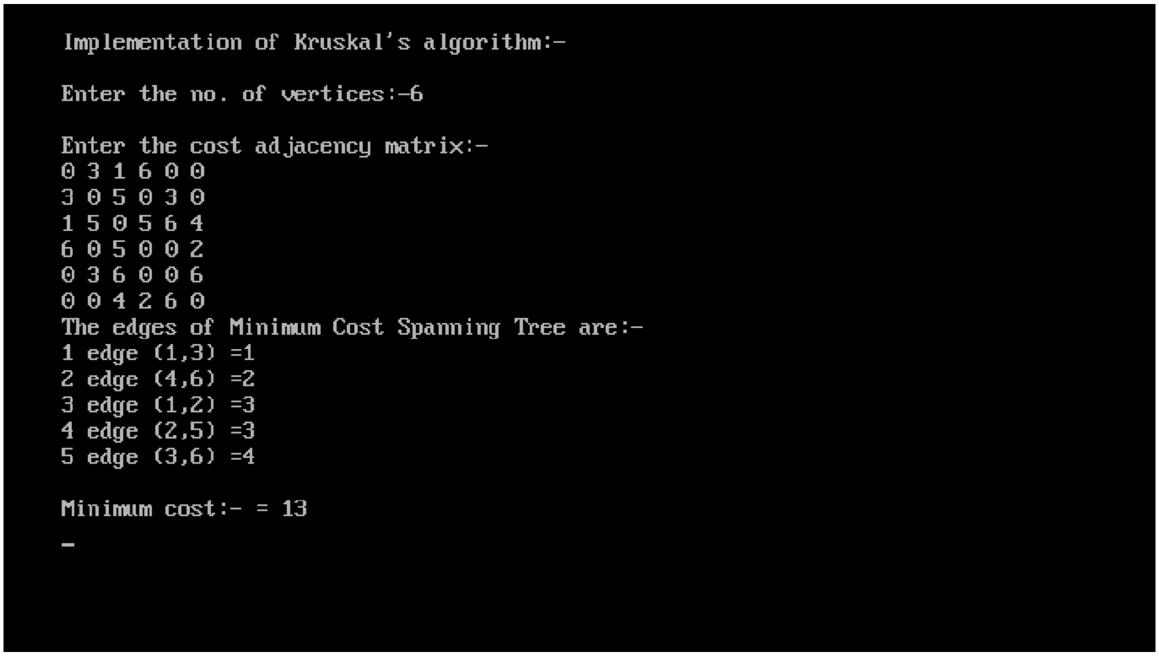
```
Implementation of Prims Algorithm:-

Enter no. of vertices:6

Enter the adjacency matrix:
0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 4
6 0 5 0 0 2
0 3 6 0 0 6
0 0 4 2 6 0

spanning tree matrix:

0       3       1       0       0       0
3       0       0       0       3       0
1       0       0       0       0       4
0       0       0       0       0       2
0       3       0       0       0       0
0       0       4       2       0       0

Total cost of spanning tree=13
```

Ques 26:- Program for MST using kruskal algorithm.

Code :-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int i,j,k,a,b,u,v,n,ne=1;
int min,mincost=0,cost[9][9],parent[9];
int find(int);
int uni(int,int);
void main()
{
        clrscr();
        printf("\nImplementation of Kruskal's algorithm:-\n");
        printf("\nEnter the no. of vertices:-");
        scanf("%d",&n);
        printf("\nEnter the cost 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;
                }
        }
        printf("The edges of Minimum Cost 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=find(u);
        v=find(v);
        if(uni(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);
    getch();
}
int find(int i)
{
    while(parent[i])
    i=parent[i];
    return i;
}
int uni(int i,int j)
{
    if(i!=j)
    {
        parent[j]=i;
        return 1;
    }
    return 0;
}
```

Output is :-



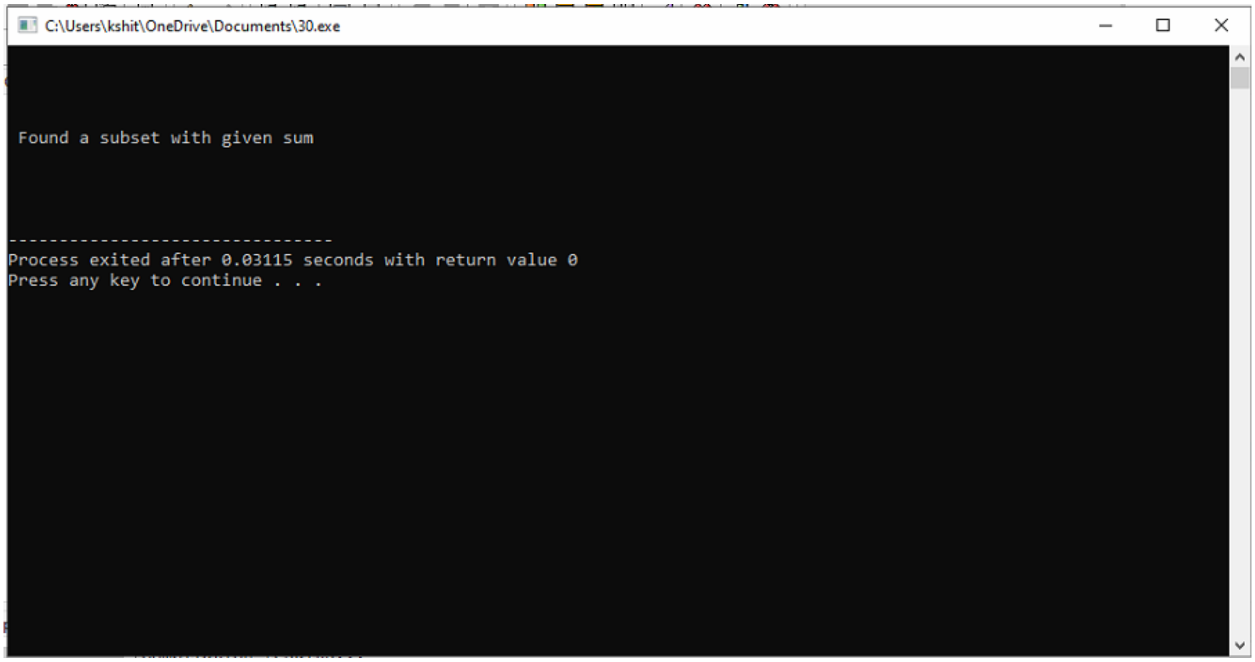
```
 Implementation of Kruskal's algorithm:-

Enter the no. of vertices:-6

Enter the cost adjacency matrix:-
0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 4
6 0 5 0 0 2
0 3 6 0 0 6
0 0 4 2 6 0
The edges of Minimum Cost Spanning Tree are:-
1 edge (1,3) =1
2 edge (4,6) =2
3 edge (1,2) =3
4 edge (2,5) =3
5 edge (3,6) =4

Minimum cost:- = 13
```

Ques 27 :- Program for Sum of subset problem.

Code :-

```
#include <stdio.h>
bool isSubsetSum(int set[], int n, int sum)
{
if (sum == 0)
return true;
if (n == 0)
return false;
if (set[n - 1] > sum)
return isSubsetSum(set, n - 1, sum);
return isSubsetSum(set, n - 1, sum)
|| isSubsetSum(set, n - 1, sum - set[n - 1]);
}
int main()
{
int set[] = { 3, 34, 4, 12, 5, 2 };
int sum = 9;
int n = sizeof(set) / sizeof(set[0]);
if (isSubsetSum(set, n, sum) == true)
printf("\n\n\n\n Found a subset with given sum\n\n\n\n");
else
printf("\n No subset with given sum");
return 0;
}
```

Output is :-



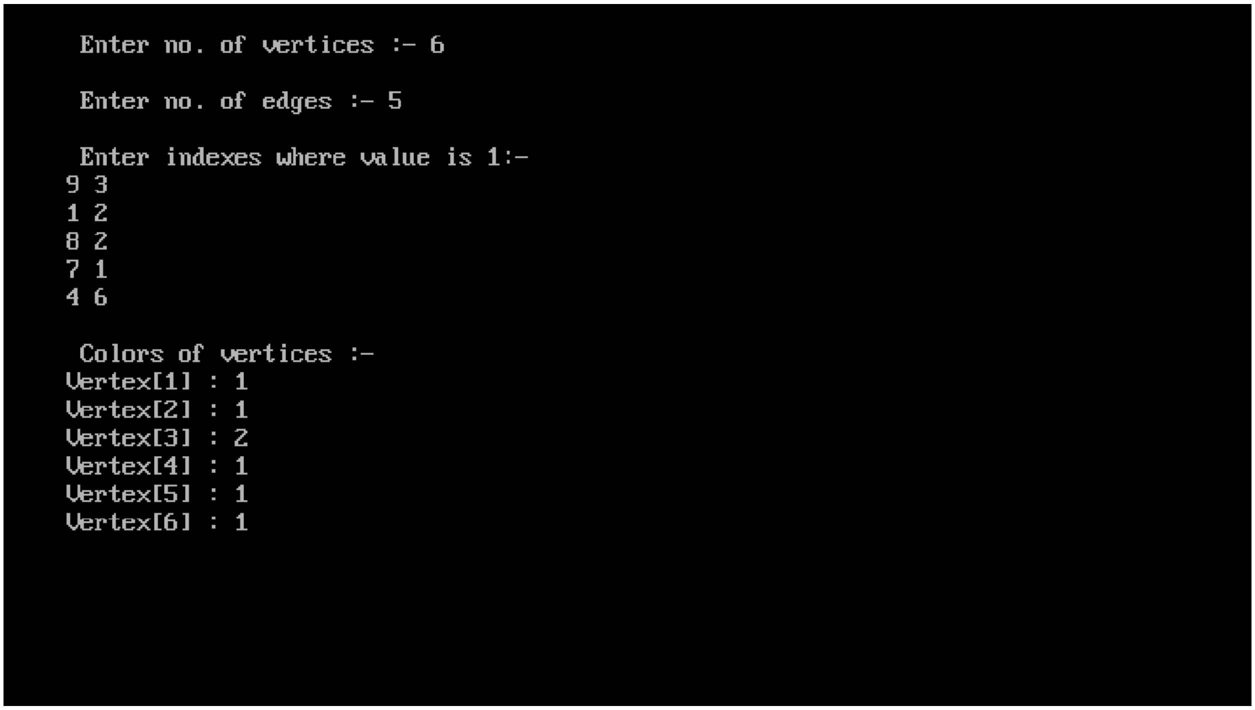
```
 Found a subset with given sum



--------------------------------
Process exited after 0.03115 seconds with return value 0
Press any key to continue . . .
```

Ques 28 :- Program for Graph coloring problem.

Code :-

```
#include<stdio.h>
#include<conio.h>
int G[50][50],x[50];
void next_color(int k)
{
   int i,j;
   x[k]=1;
   for(i=0;i<k;i++)
{
     if(G[i][k]!=0 && x[k]==x[i])
       x[k]=x[i]+1;
   }
}

void main()
{
  clrscr();
  int n,a,i,j,k,l;
  printf("\n Enter no. of vertices :- ");
  scanf("%d",&n);
  printf("\n Enter no. of edges :- ");
  scanf("%d",&a);

  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
      G[i][j]=0;

  printf("\n Enter indexes where value is 1:- \n");
  for(i=0;i<a;i++)
{
    scanf("%d %d",&k,&l);
    G[k][l]=1;
    G[l][k]=1;
  }
  for(i=0;i<n;i++)
    next_color(i);

  printf("\n Colors of vertices :-\n");
  for(i=0;i<n;i++)
    printf("Vertex[%d] : %d\n",i+1,x[i]);

  getch();
}
```

Output is :-



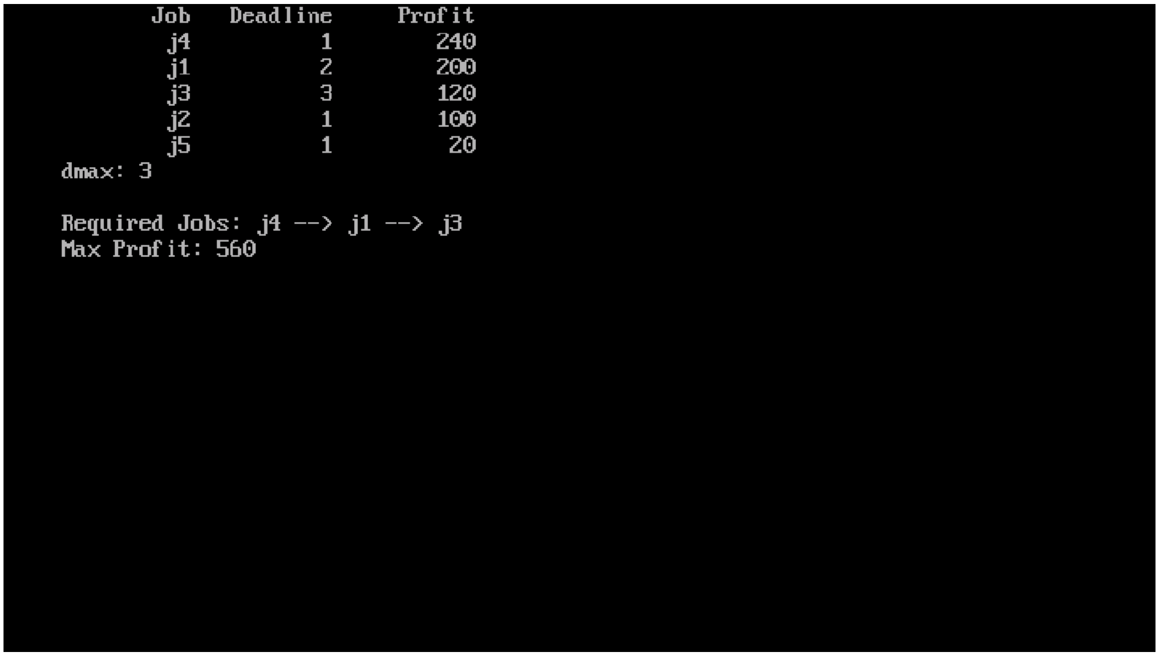
```
     Enter no. of vertices :- 6

     Enter no. of edges :- 5

     Enter indexes where value is 1:-
    9 3
    1 2
    8 2
    7 1
    4 6

     Colors of vertices :-
    Vertex[1] : 1
    Vertex[2] : 1
    Vertex[3] : 2
    Vertex[4] : 1
    Vertex[5] : 1
    Vertex[6] : 1
```

Ques 29 :- Program for Job scheduling with deadline problem.

Code :-

```
#include <stdio.h>
#include <conio.h>
#define MAX 100

typedef struct Job
 {
  char id[5];
  int deadline;
  int profit;
}
 Job;

void jobSequencingWithDeadline(Job
jobs[], int n);
int minValue(int x, int y) {
  if(x < y) return x;
  return y;
}
int main(void) {
clrscr();
  int i, j;
  Job jobs[5] = {

    {"j1", 2, 200},
    {"j2", 1, 100},
    {"j3", 3, 120},
    {"j4", 1, 240},
    {"j5", 1,  20},
  };
  Job temp;
  int n = 5;
 for(i = 1; i < n; i++) {
    for(j = 0; j < n - i; j++) {
      if(jobs[j+1].profit > jobs
[j].profit) {
            temp = jobs[j+1];
            jobs[j+1] = jobs[j];
            jobs[j] = temp;
      }
    }
  }
  printf("%10s %10s %10s\n", "Job", "Deadline", "Profit");
  for(i = 0; i < n; i++)
   {
    printf("%10s %10i %10i\n", jobs[i].id, jobs[i].deadline, jobs[i].profit);
  }
  jobSequencingWithDeadline(jobs, n);
  return 0;
}
void jobSequencingWithDeadline(Job
jobs[], int n) {
  int i, j, k, maxprofit;
  int timeslot[MAX];
  int filledTimeSlot = 0;
  int dmax = 0;
  for(i = 0; i < n; i++) {
    if(jobs[i].deadline > dmax) {
      dmax = jobs[i].deadline;
    }
  }
  for(i = 1; i <= dmax; i++)
   {
    timeslot[i] = -1;
  }
  printf("dmax: %d\n", dmax);
  for(i = 1; i <= n; i++)
   {
    k = minValue(dmax, jobs[i - 1].deadline);
    while(k >= 1)
     {
      if(timeslot[k] == -1)
       {
            timeslot[k] = i-1;
            filledTimeSlot++;
            break;
      }
      k--;
    }
    if(filledTimeSlot == dmax) {
      break;
    }
  }
  printf("\nRequired Jobs: ");
  for(i = 1; i <= dmax; i++) {
    printf("%s", jobs[timeslot[i]].id);
    if(i < dmax) {
      printf(" --> ");
    }
  }
  maxprofit = 0;
  for(i = 1; i <= dmax; i++)
 {
    maxprofit += jobs[timeslot[i]].profit;
  }
  printf("\nMax Profit: %d\n", maxprofit);
}
```

Output is :-



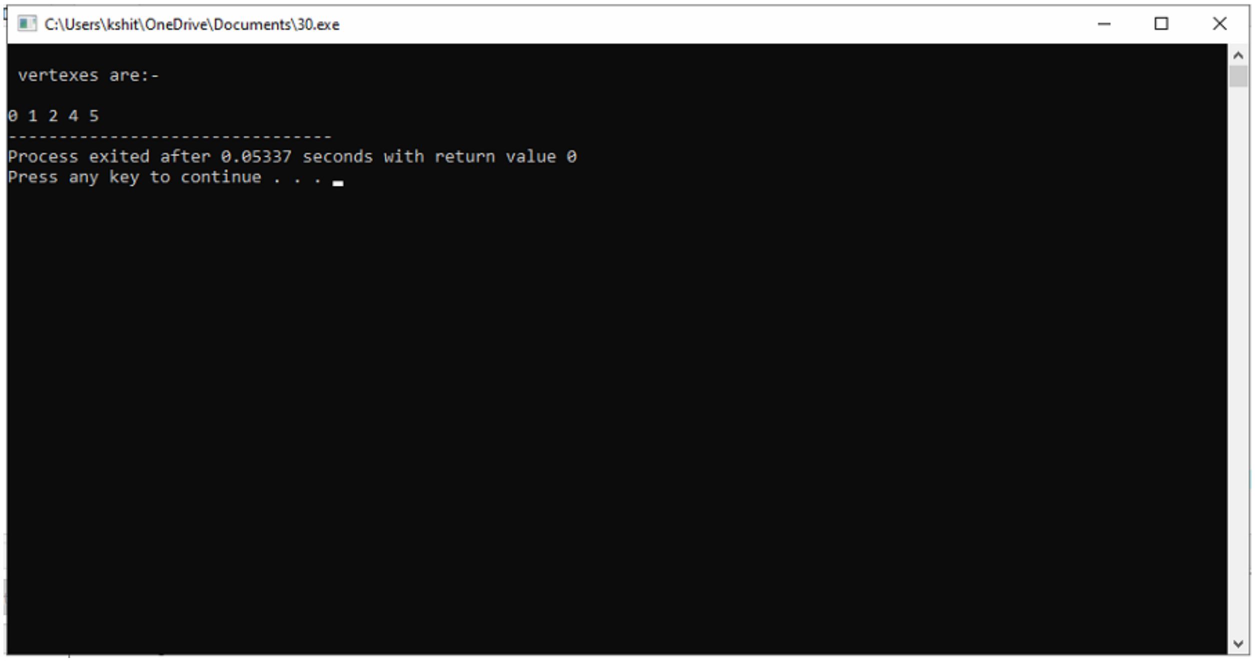
```
            Job   Deadline     Profit
             j4          1        240
             j1          2        200
             j3          3        120
             j2          1        100
             j5          1         20
     dmax: 3

     Required Jobs: j4 --> j1 --> j3
     Max Profit: 560
```

Ques 30 :- Program for Vertex cover problem using approximation algorithm.

Code :-

```
#include<iostream>
#include <list>
using namespace std;
class Graph
{
int V;
list<int> *adj;
public:
Graph(int V);
void addEdge(int v, int w);
void printVertexCover();
};
Graph::Graph(int V)
{
this->V = V;
adj = new list<int>[V];
}
void Graph::addEdge(int v, int w)
{
adj[v].push_back(w);
adj[w].push_back(v);
}
void Graph::printVertexCover()
{
bool visited[V];
for (int i=0; i<V; i++)
visited[i] = false;
list<int>::iterator i;
for (int u=0; u<V; u++)
{
if (visited[u] == false)
{
for (i= adj[u].begin(); i != adj[u].end(); ++i)
{
int v = *i;
if (visited[v] == false)
{
visited[v] = true;
visited[u] = true;
break;
}
}
}
}
for (int i=0; i<V; i++)
if (visited[i])
cout << i << " ";
}
int main()
{
cout << "\n vertexes are:-\n  "<< endl;
Graph g(7);
g.addEdge(5, 1);
g.addEdge(0, 2);
g.addEdge(1, 5);
g.addEdge(1, 7);
g.addEdge(4, 5);
g.addEdge(4, 9);
g.printVertexCover();
return 0;
}
```

Output is :-



```
 vertexes are:-

0 1 2 4 5
--------------------------------
Process exited after 0.05337 seconds with return value 0
Press any key to continue . . .
```