1.// Program to implement Selection sort

```
#include<stdio.h>
void main()
{
int arr[20],n,i,j,min_indx,temp;
printf("Enter size of the array: ");
scanf("%d",&n);
printf("Enter %d elements: ",n);
for(i=0;i<n;i++)
scanf("%d",&arr[i]);
for (i = 0; i < n; i++)
{
// Find the minimum element in unsorted array
min_indx = i;
for (j = i+1; j < n; j++)
if (arr[j] < arr[min_indx])
min_indx = j;
// Swap the found minimum element with the ith element
temp=arr[min_indx];
arr[min_indx]=arr[i];
arr[i]=temp;
}
printf("\n Sorted array is ");
for(i=0;i<n;i++)
printf(" %d",arr[i]);
}
```

```
// Program to implement Insertion sort
#include<stdio.h>
void main()
{
int arr[20],n,i,j,key;
printf("Enter size of the array: ");
scanf("%d",&n);
printf("Enter %d elements: ",n);
for(i=0;i<n;i++)
scanf("%d",&arr[i]);
for (int i = 1; i < n; i++)
{
key = arr[i];
j = i - 1;
while (j >= 0 && arr[j] > key)
{
arr[j + 1] = arr[j];
j = j - 1;
}
arr[j + 1] = key;
}
printf("\n Sorted array is ");
for(i=0;i<n;i++)
printf(" %d",arr[i]);
}
```

```
// Program to implement Merge sort
#include<stdio.h>
void mergesort(int a[10], int low, int high)
{
int mid;
if(low<high)
{
mid=(low+high)/2;
mergesort(a,low,mid);
mergesort(a,mid+1,high);
combine(a,low,mid,high);
}
}
void combine(int a[10],int low,int mid, int high)
{
int i,j,k,temp[10];
i=low;
j=mid+1;
k=low;
while(i<=mid && j<=high)
{
if(a[i]<a[j])
{
temp[k]=a[i];
k++; i++;
}else{
temp[k]=a[j];
k++; j++;
}
}while(i<=mid){
temp[k]=a[i];
```

```
k++;
i++;
}
while(j<=high)

{
temp[k]=a[j];
k++;
j++;
}
for(i=low;i<=high;i++)
a[i]=temp[i];
}
void main()
{
int a[10],n,i;
printf("\n Enter no of elements ");
scanf("%d",&n);
printf("\n Enter the elements\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
mergesort(a,0,n-1);
printf("\nSorted array is \n");
for(i=0;i<n;i++)
printf("%d ",a[i]);
}
```

```
// Program to implement Quick sort
#include<stdio.h>
void quicksort(int [10],int,int);
void main(){
int a[20],n,i;
printf("Enter size of the array: ");
scanf("%d",&n);
printf("Enter %d elements: ",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
quicksort(a,0,n-1);
printf("Sorted elements: ");
for(i=0;i<n;i++)
printf("%d ",a[i]);
}
void quicksort(int a[10],int first,int last){
int pivot,j,temp,i;
if(first<last){
pivot=first;
i=first;
j=last;
while(i<j){
while(a[i]<=a[pivot]&&i<last)
i++;
while(a[j]>a[pivot]&&j>first)
j--;
if(i<j){
temp=a[i];a[i]=a[j];
a[j]=temp;
}
}
```

```
temp=a[pivot];
a[pivot]=a[j];
a[j]=temp;
quicksort(a,first,j-1);
quicksort(a,j+1,last);
}
}
```

```
/* Program to implement binary search */
#include <stdio.h>
void main(){
int a[10],n,i, first, last, middle,search;
printf("Enter number of elements\n");
scanf("%d",&n);
printf("Enter the elements");
for (i=0;i<n;i++ )
scanf("%d",&a[i]);
printf("Enter element to be searched :");
scanf("%d",&search);
first=0;
last=n-1;
middle=(first+last)/2;
while(first<=last)
{
if(a[middle]==search)
break;
else if(a[middle] < search )
first = middle + 1;
else
last = middle - 1;
middle=(first+last)/2;
}
if (first>last)
printf("\n Element not found");
else
printf("\n Element found");
}
```

```
// Program for Strassen's matrix multiplication
#include<stdio.h>
void main()
{
int Z[2][2];
int i, j;
int A,B,C,D,E,F,G,H;
int p1, p2, p3, p4 , p5, p6, p7;
int c1, c2, c3, c4;
int X[2][2] = { {1, 2},
{3, 4} };
int Y[2][2] = { {5, 6},
{7, 8} };
printf("The first matrix is: ");
for(i = 0; i < 2; i++)
{
printf("\n");
for(j = 0; j < 2; j++)
printf("%d\t", X[i][j]);
}
printf("\nThe second matrix is: ");
for(i = 0; i < 2; i++)
{
printf("\n");
for(j = 0; j < 2; j++)
printf("%d\t", Y[i][j]);
}
A=X[0][0];
B=X[0][1];
C=X[1][0];
D=X[1][1];
```

```
E=Y[0][0];
F=Y[0][1];
G=Y[1][0];
H=Y[1][1];
p1 = A*(F-H);
p2 = H*(A+B);
p3 = E*(C+D);
p4 = D*(G-E);
p5 = (A+D)*(E+H);
p6 = (B-D)*(G+H);
p7 = (A-C)*(E+F);
c1 = p4+p5+p6-p2;

c2 = p1+p2;
c3 = p3+p4;
c4 = p1-p3+p5-p7;
Z[0][0]=c1;
Z[0][1]=c2;
Z[1][0]=c3;
Z[1][1]=c4;
printf("\nProduct achieved using Strassen's algorithm: ");
for(i = 0; i < 2 ; i++)
{
printf("\n");
for(j = 0; j < 2; j++)
printf("%d\t", Z[i][j]);
}
}
```

```
// Program to implement Fractional knapsack
# include<stdio.h>
void main()
{
float weight[20], profit[20], capacity;
int n, i, j;
float ratio[20], fract=1.0, tp=0, temp;
printf("\nEnter the capacity of knapsack:- ");
scanf("%f",&capacity);
printf("\nEnter the no. of items:- ");
scanf("%d",&n);
printf("\nEnter the weights and profits of each item:- ");
for (i = 0; i < n; i++)
scanf("%f %f",&weight[i],&profit[i]);
for (i = 0; i < n; i++)
ratio[i] = profit[i] / weight[i];
for (i = 0; i < n; i++)
{
for (j = i + 1; j < n; 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;
```

```
}
}
}
for (i = 0; i < n; i++)
{
if (weight[i] > capacity)
break;
else
{
capacity = capacity - weight[i];

tp = tp + profit[i];
}
}
printf("%f ",tp);
if (i < n)
fract = capacity / weight[i];
tp = tp + (fract * profit[i]);
printf("\nMaximum profit is:- %f ", tp);
}
```

```
// Program to find all pair shortest path using Floyd-Warshall algorithm
#include <stdio.h>
#define n 4
#define INF 999
int main(){
int A[n][n] = {{0, 5, 9, INF},
{INF, 0, 1, INF},
{INF, INF, 0, 2},
{INF, 3, INF, 0}};
int i, j, k;
for (k = 0; k < n; k++){
        for (i = 0; i < n; i++){
                for (j = 0; j < n; j++){
if (A[i][k] + A[k][j] < A[i][j])
A[i][j] = A[i][k] + A[k][j];}}}
printf("\n The shortest distances from every pair of vertices are:\n");
for (i = 0; i < n; i++){
for (j = 0; j < n; j++){
if (A[i][j] == INF)
printf("%s\t", "INF");
else
printf("%d\t", A[i][j]);
}
printf("\n");
}
}
```

```
// Program to Find the longest common subsequence
#include<stdio.h>
#include<string.h>
char x[20], y[20], b[20][20];
void print_lcs(int i, int j)
{
if(i==0 || j==0)
return;
if(b[i][j]=='c')
{
print_lcs(i-1,j-1);
printf("%c",x[i-1]);
}
else if(b[i][j]=='u')
print_lcs(i-1,j);
else
print_lcs(i,j-1);
}
void main()
{
int i,j,m,n,c[20][20];
printf("\nEnter the 1st sequence: ");
scanf("%s",x);
printf("\nEnter the 2nd sequence: ");
scanf("%s",y);
printf("\nThe Longest Common Subsequence is ");
m=strlen(x);
n=strlen(y);
for(i=0;i<=m;i++)
c[i][0]=0;
for(i=0;i<=n;i++)
```

```
c[0][i]=0;
//c, u and l denotes cross, upward and leftward directions respectively
for(i=1;i<=m;i++)
{
for(j=1;j<=n;j++)

{
if(x[i-1]==y[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';
}
}
}
print_lcs(m,n);
```