Insertion Sort

```
#include<stdio.h>
int main(){
   int i,j,n,key,A[20];
   printf("Enter No of Elements : \n");
   scanf("%d",&n);
   printf("Enter Array Elements : \n");
   for(i=0;i<n;i++)
   scanf("%d",&A[i]);
   printf("Display Elements Before Sorting : \n");
   for(i=0;i<n;i++)
   printf("%d \t",A[i]);
   //Sorting
   for(i=1;i<n;i++){
      key=A[i];
      j=i-1;
   while(j>=0 && key<=A[j]) {
      A[j+1]=A[j];
      j=j-1;
   }
   A[j+1]=key;
   }
   //Done
   printf("\nDisplay Elements After Sorting : \n");
   for(i=0;i<n;i++)
   printf("%d \t",A[i]);
   return 0;
 }
```

Selection Sort

```
#include<stdio.h>
int main(){
   int i,j,n,index,swap,A[20];
   printf("Enter NO of Elements : \n");
   scanf("%d",&n);
   printf("Enter Array Elements : \n");
   for(i=0;i<n;i++)
   scanf("%d",&A[i]);
   printf("Display Elements Before Sorting : \n");
   for(i=0;i<n;i++)
   printf("%d\t",A[i]);
   //Sorting
   for(i = 0; i < (n - 1); i++)
        {
           index=i;
           for(j = i + 1; j < n; j++)
        {
           if(A[index]>A[j])
           index=j;
        }
           if(index != i)
        {
           swap=A[i];
           A[i]=A[index];
           A[index]=swap;
        }
        }
           //done
           printf("\nDisplay Elements After Sorting : \n");
           for(i=0;i<n;i++)
           printf("%d \t",A[i]);
        return 0;
        }
```

Merge Sort

```
#include<stdio.h>
void Merge(int A[],int B,int mid,int E){
   int i,j,k;
   int n1=mid - B + 1;
   int n2= E - mid;
   int LA[n1],RA[n2];
   for(i=0;i<n1;i++)
   LA[i]=A[B + i];
   for(j=0;j<n2;j++)
   RA[j]=A[mid + 1 + j];
   i=0;
   j=0;
   k=B;
   while(i<n1 && j<n2){
      if(LA[i]<RA[j]){
         A[k]=LA[i];
         i++;
      }
      else
      {
         A[k]=RA[j];
         j++;
      }
      k++;
   }
   while(i<n1){
```

```
        A[k]=LA[i];
        i++;
        k++;
    }
    while(j<n2){
        A[k]=RA[j];
        j++;
        k++;
    }
}
void MergeSort(int A[],int B,int E){
    if(B<E){
        int mid=(B + E)/2;
        MergeSort(A,B,mid);
        MergeSort(A,mid + 1,E);
        Merge(A,B,mid,E);
    }
}
int main(){
    int i,j,A[20],n;
    printf("Enter NO of Elements : ");
    scanf("%d",&n);
    printf("Enter Array Elements : ");
    for(i=0;i<n;i++)
    scanf("%d",&A[i]);
    printf("Displaying Elements Before sorting: \n");
    for(i=0;i<n;i++)
```

```
    printf("%d\t",A[i]);
    MergeSort(A,0,n - 1);
    printf("\nDisplaying Elements After sorting: \n");
    for(i=0;i<n;i++)
    printf("%d\t",A[i]);
    return 0;
}
```

Quick Sort

```
 #include<stdio.h>
int Partation(int A[],int S,int E){
   int i,j,pivot,Key;
   pivot=A[S];
   i=S + 1;
   j=E;
   do{
      while(A[i]<=pivot){
         i++;
      }
      while(A[j]>pivot){
         j--;
      }
      if(i<j){
         Key=A[i];
         A[i]=A[j];
         A[j]=Key;
      }
   }while(i<j);
   Key=A[S];
   A[S]=A[j];
   A[j]=Key;

return j;
}
void QuickSort(int A[],int S,int E){
   int PartationIndex;
   if(S<E){
      PartationIndex=Partation(A,S,E);
      QuickSort(A,S,PartationIndex - 1);
      QuickSort(A,PartationIndex + 1,E);
   }
}

int main(){
   int A[20],n,i;
   printf("Enter No of Elements : ");
   scanf("%d",&n);
```

```
    printf("Enter Array Elements : \n");
    for(i=0;i<n;i++)
    scanf("%d",&A[i]);
    printf("Display Elements before sorting : \n");
    for(i=0;i<n;i++)
    printf("%d\t",A[i]);
    QuickSort(A,0,n - 1);
    printf("Display Elements before sorting : \n");
    for(i=0;i<n;i++)
    printf("%d\t",A[i]);
    return 0;
}
```

LCS

```
#include<stdio.h>
#include<string.h>
void lcsalgo(char s1[],char s2[],int m,int n){
   int i,j,lcst[20][20];
   char lcsd[20][20];
   for(i=0;i<=m;i++){
   lcst[i][0]=0;
   lcsd[i][0]='-';
   }
   for(j=0;j<=n;j++){
   lcst[0][j]=0;
   lcsd[0][j]='-';
   }
   for(i=1;i<=m;i++){
   for(j=1;j<=n;j++){
   if(s1[i - 1]==s2[j - 1]){
      lcst[i][j]=1 + lcst[i - 1][j - 1];
      lcsd[i][j]='d';
   }
   else
   if(lcst[i - 1][j]>=lcst[i][j - 1]){
   lcst[i][j]=lcst[i - 1][j];
   lcsd[i][j]='u';
   }
   else{
   lcst[i][j]=lcst[i][j - 1];
   lcsd[i][j]='l';
   }
   }
}
   int index=lcst[m][n];
   char lsca[index + 1];
   lsca[index]='\0';
   i=m;
   j=n;
   while(i>0 && j>0){
      if(s1[i - 1]==s2[j - 1]){
         lsca[index - 1]=s1[i - 1];
```

```
            i--;
            j--;
            index--;
        }
        else
        if(lcst[i- 1][j]>lcst[i][j - 1])
          i--;
        else
        j--;
    }
    printf("Table 1 \n");
    for(i=0;i<=m;i++){
    for(j=0;j<=n;j++){
        printf("%d \t",lcst[i][j]);
    }
    printf("\n");
}
    printf("Table 2 \n");
    for(i=0;i<=m;i++){
    for(j=0;j<=n;j++){
    printf("%c \t",lcsd[i][j]);
    }
    printf("\n");
    }
    printf(" LCS : %s",lsca);


}
int main(){
    int i,j,m,n;
    char s1[20],s2[20];
    printf("Enter String 1 ");
    scanf("%[^\n]%*c",s1);
    printf("Enter String 2 ");
    scanf("%[^\n]%*c",s2);
    m=strlen(s1);
    n=strlen(s2);
    lcsalgo(s1,s2,m,n);
return 0;
}
```

Floyd Warshall

```
#include<stdio.h>
int min(int a,int b) {
    if(a<b)
     return(a); else
     return(b);
}
void main() {
    int p[10][10],w,n,e,u,v,i,j,k;
    printf("\n Enter the number of vertices:");
    scanf("%d",&n);
    printf("\n Enter the number of edges:\n");
    scanf("%d",&e);
    for (i=1;i<=n;i++) {
            for (j=1;j<=n;j++)
            if(i==j)
            p[i][j]=0;
            else
              p[i][j]=999;
    }
    for (i=1;i<=e;i++) {
            printf("\n Enter the end vertices of edge%d with its weight \n",i);
            scanf("%d%d%d",&u,&v,&w);
            p[u][v]=w;
    }
    printf("\n GIven data : \n");
```

```
    for (i=1;i<=n;i++) {
            for (j=1;j<=n;j++)
              printf("%d \t",p[i][j]);
            printf("\n");
    }
for (k=1;k<=n;k++)
      for (i=1;i<=n;i++)
       for (j=1;j<=n;j++)
        if(i!=j)
          p[i][j]=min(p[i][j],p[i][k]+p[k][j]);
    printf("\n Resultant Matrix \n");
    for (i=1;i<=n;i++) {
            for (j=1;j<=n;j++)
              printf("%d \t",p[i][j]);
            printf("\n");
    }
    printf("\n The shortest paths are:\n");
    for (i=1;i<=n;i++)
      for (j=1;j<=n;j++) {
            if(i!=j)
               printf("\n <%d,%d>=%d",i,j,p[i][j]);
    }

}
```

Fractional Knapsack

```
#include<stdio.h>
void knapsack(int n,float w[],float v[], float c){
   int i,j,u;
   float x[10],p=0;
   u=c;
   for(i=0;i<n;i++)
   x[i]=0.0;
   for(i=0;i<n;i++){
      if(w[i]>u)
      break;
      else
      {
         x[i]=1.0;
         p=p+v[i];
         u=u-w[i];
   }
   }
   if(i<n)
      x[i]=u/w[i];
   p=p+(x[i]*v[i]);
   printf("Solution vector : \t");
   for(i=0;i<n;i++)
   printf("%f \t",x[i]);
   printf("\nProft = %f",p);
}
int main(){
```

```
float w[10],v[10],r[10],c;
int i,j,t,n;
printf("Enter no of objects : ");
scanf("%d",&n);
printf("Enter Values : \n");
for(i=0;i<n;i++)
scanf("%f",&v[i]);
printf("Enter Weights : \n");
for(i=0;i<n;i++)
scanf("%f",&w[i]);
printf("Enter capacity of knapsack : ");
scanf("%f",&c);
for(i=0;i<n;i++)
r[i]=v[i]/w[i];
for(i=0;i<n;i++){
for(j=i+ 1;j<n;j++){
   if(r[i]<r[j]){
   t=r[j];
   r[j]=r[i];
   r[i]=t;

   t=w[j];
   w[j]=w[i];
   w[i]=t;

   t=v[j];
   v[j]=v[i];
```

```
        v[i]=t;
    }
    }
}
    knapsack(n,w,v,c);

    return 0;
}
```