1. kruskal

```
#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[10][10],parent[10];
int find(int);
int uni(int,int);

void main()
{
 //clrscr();
 printf("\n\n\tImplementation of Kruskal's algorithm\n\n");
 printf("\nEnter the no. of vertices\n");
 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("\nThe edges of Minimum Cost Spanning Tree are\n\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("\n%d edge (%d,%d) =%d\n",ne++,a,b,min);
   mincost +=min;
  }
  cost[a][b]=cost[b][a]=999;
 }
 printf("\n\tMinimum 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;
}
```

2.dijkshtra

```
#include "stdio.h"
#include "conio.h"
#define infinity 999

void dij(int n,int v,int cost[10][10],int dist[])
{
 int i,u,count,w,flag[10],min;
 for(i=1;i<=n;i++)
  flag[i]=0,dist[i]=cost[v][i];
 count=2;
 while(count<=n)
 {
  min=99;
  for(w=1;w<=n;w++)
   if(dist[w]<min && !flag[w])
    min=dist[w],u=w;
  flag[u]=1;
  count++;
  for(w=1;w<=n;w++)
   if((dist[u]+cost[u][w]<dist[w]) && !flag[w])
    dist[w]=dist[u]+cost[u][w];
 }
}

void main()
{
 int n,v,i,j,cost[10][10],dist[10];
 //clrscr();
 printf("\n Enter the number of nodes:");
 scanf("%d",&n);
 printf("\n Enter the cost 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]=infinity;
  }
 printf("\n Enter the source matrix:");
 scanf("%d",&v);
 dij(n,v,cost,dist);
 printf("\n Shortest path:\n");
 for(i=1;i<=n;i++)
  if(i!=v)
   printf("%d->%d,cost=%d\n",v,i,dist[i]);
 getch();
}
```

3.Longest Common Subsequence

```
/* Dynamic Programming implementation of LCS problem */
#include<iostream>
#include<cstring>
#include<cstdlib>
using namespace std;

/* Returns length of LCS for X[0..m-1], Y[0..n-1] */
void lcs( char *X, char *Y, int m, int n )
{
   int L[m+1][n+1];

   /* Following steps build L[m+1][n+1] in bottom up fashion. Note
      that L[i][j] contains length of LCS of X[0..i-1] and Y[0..j-1] */
```

```
  for (int i=0; i<=m; i++)
  {
   for (int j=0; j<=n; j++)
   {
     if (i == 0 || j == 0)
       L[i][j] = 0;
     else if (X[i-1] == Y[j-1])
       L[i][j] = L[i-1][j-1] + 1;
     else
       L[i][j] = max(L[i-1][j], L[i][j-1]);
   }
  }

  // Following code is used to print LCS
  int index = L[m][n];

  // Create a character array to store the lcs string
  char lcs[index+1];
  lcs[index] = '\0'; // Set the terminating character

  // Start from the right-most-bottom-most corner and
  // one by one store characters in lcs[]
  int i = m, j = n;
  while (i > 0 && j > 0)
  {
    // If current character in X[] and Y are same, then
    // current character is part of LCS
    if (X[i-1] == Y[j-1])
    {
       lcs[index-1] = X[i-1]; // Put current character in result
       i--; j--; index--;     // reduce values of i, j and index
    }

    // If not same, then find the larger of two and
    // go in the direction of larger value
    else if (L[i-1][j] > L[i][j-1])
      i--;
    else
      j--;
  }

  // Print the lcs
  cout << "LCS of " << X << " and " << Y << " is " << lcs;
}

/* Driver program to test above function */
int main()
{
  char X[] = "AGGTAB";
  char Y[] = "GXTXAYB";
  int m = strlen(X);
  int n = strlen(Y);
  lcs(X, Y, m, n);
  return 0;
}
```

4.prims

```
#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];
void main()
{
        //clrscr();
        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);
    getch();
}
```

5. RADIX SORT

```
// fundamentals headers
#include<iostream>
#include<algorithm>
#include<cstdlib>
#include<cstdio>
#include<cmath>

using namespace std;

int main()
{
    int n,k=0,maxi = 0;
    cout<<"Enter the number of element:"<<endl;
    cin>>n;
    int a[n];
    for (int i = 0; i < n; ++i)
    {
        cin>>a[i];
        if (k < a[i])
        {
            k = a[i];
        }
    }

    while(k > 0)
    {
        k = k/10;
        maxi++;
    }
    //cout<<" max : "<<maxi<<endl;

    for (int i = 0; i < maxi; ++i)
    {
        for (int j = 0; j < n-1; ++j)
        {
            int copiy = a[j]/pow(10,i);
            int temp = copiy % 10;
            for (int k = j+1; k < n; ++k)
            {
```

```
                                    int copy_k = a[k]/pow(10,i);
                                    if(temp > copy_k % 10 )
                                          {
                                             swap(a[k],a[j]);
                                          }
                                }
                        }
            }

            for (int i = 0; i < n; ++i)
            {
                        cout<<" "<<a[i];
            }
            return 0;
}
```

6. COUNTING SORT

```
// fundamentals headers
#include<iostream>
#include<algorithm>
#include<cstdlib>
#include<cstdio>
#include<cmath>

using namespace std;

int main()
{
            int n,k=0;
            cout<<"Enter the number of element:"<<endl;
            cin>>n;
            int a[n];
            for (int i = 0; i < n; ++i)
            {
                        cin>>a[i];
                        if (k < a[i])
                        {
                                    k = a[i];
                        }
            }
            int c[k+1];
   for (int j = 0; j < k+1; ++j)
            {
                        c[j] = 0;
            }
            for (int j = 0; j < n; ++j)
```

```
    {
        c[a[j]] = c[a[j]]+1;
        cout<<" "<<c[a[j]];
    }
    cout<<endl;
    int pos=0;
    for (int i = 0; i < k+1; ++i)
    {
        while(c[i] > 0)
        {
            a[pos] = i; pos++;
            c[i]--;
        }
    }
    for (int j = 0; j < n; ++j)
    {
        cout<<" "<<a[j];
    }

    return 0;
}
```

7.BINARY SEARCH

```
#include<iostream.h>
#include<conio.h>
int binary_search( int array[], int first, int last, int value);
int main()
{
int list[10],x;    cin>>x;

 for (int k=0; k<10; k++)
     cin>>list[k];
 cout<< "binary search results: "<< binary_search(list,1,10,x)<<endl;
 return 0;
 getch();
}                          //end of main

int binary_search(int array[],int first,int last, int search_key)
{
 int index;
     if (first > last)
         index = -1;
     else
     {
        int mid = (first + last)/2;
```

```
                        if (search_key == array[mid])
                        index = mid;
                        else
                       {
                                if (search_key < array[mid])
                                index = binary_search(array,first, mid-1, search_key);
                                else
                                index = binary_search(array, mid+1, last, search_key);
                       }

            }                                   // end if
             return index;
 }                                              // end binarySearch
```

8.Quick Sort using Divide and Conquer*/

```
#include<stdio.h>
#include<conio.h>
int arr[40];
void quicksort(int a[],int p,int r);
int partition(int a[],int p,int r);
void exchange(int i,int j);
void quicksort(int a[],int p,int r)
{
            int q;
            if(p<r){
              q=partition(a,p,r);
              quicksort(a,p,q-1);
              quicksort(a,q+1,r);
               }
  }
int partition(int a[],int p,int r){
            int x,j,i;
            x=a[r];
            i=p-1;
            for(j=p;j<=(r-1);j++)
               if(a[j]<x){
                      i=i+1;
                      exchange(i,j);
                }
               exchange(i+1,r);
       return(i+1);
    }
void exchange(int i,int j){
    int temp;
    temp=arr[i];
    arr[i]=arr[j];
    arr[j]=temp;
    }
int main(){
    int n,i;
    printf("\nEnter no . elements needed : ");
    scanf("%d",&n);
           printf("\nEnter elements : ");
    for(i=1;i<=n;i++)
        scanf("%d",&arr[i]);
    quicksort(arr,1,n);
    printf("\nSorted Array is : ");
    for(i=1;i<=n;i++)
         printf("%4d",arr[i]);
    getch();
    return(0);
   }
```

9.MAX_MIN USING DIVIDE & CONQUER*/

```
#include<stdio.h>
int max, min;
int a[100];
void maxmin(int i, int j)
{
        int max1, min1, mid;
        if(i==j)
                 max = min = a[i];


        else
        {
                if(i == j-1)
                {
                        if(a[i] <a[j])
                        {
                                max = a[j];
                                min = a[i];
                        }
                        else
                        {
                                max = a[i];
                                min = a[j];
                        }
                }
                else
                {
                        mid = (i+j)/2;
                        maxmin(i, mid);
                        max1 = max;    min1 = min;
                        maxmin(mid+1, j);
                        if(max <max1)
                                max = max1;
                        if(min > min1)
                                min = min1;
                }
        }
}
void main ()
{
        int i, num;
        clrscr();
        printf ("\n\t\t\tMAXIMUM & MINIMUM\n\n");
        printf ("\nEnter the total number of numbers : ");
        scanf ("%d",&num);
        printf ("Enter the numbers : \n");
        for (i=1;i<=num;i++)
        {
                scanf ("%d",&a[i]);
        }

        max = a[0];
        min = a[0];
        maxmin(1, num);
        printf ("Maximum element in an array : %d\n", max);
        printf ("Minimum element in an array : %d\n", min);
        getch();
}
```

10.MERGE SORT

```
#include<stdio.h>
#include<conio.h>
int a[50];
void merge(int,int,int);
void merge_sort(int low,int high)
{
         int mid;
         if(low<high)
         {
                  mid=(low+high)/2;
                  merge_sort(low,mid);
                  merge_sort(mid+1,high);
                merge(low,mid,high);
         }
}
void merge(int low,int mid,int high)
{
 int h,i,j,b[50],k;
 h=low;
 i=low;
 j=mid+1;
 while((h<=mid)&&(j<=high))
 {
          if(a[h]<=a[j])
          {
```

```
                          b[i]=a[h];
                           h++;
             }
              else
              {
                           b[i]=a[j];
                          j++;
              }
             i++;
 }
 if(h>mid)
 {
             for(k=j;k<=high;k++)
             {
                         b[i]=a[k];
                         i++;
             }
 }
 else
 {
            for(k=h;k<=mid;k++)
             {
                          b[i]=a[k];
                         i++;
            }
 }
 for(k=low;k<=high;k++)
             a[k]=b[k];
}

int main()
{
 int num,i;

printf("\t\t\tMERGE SORT\n");
 printf("\nEnter the total numbers: ");
 scanf("%d",&num);
 printf("\nEnter %d numbers: \n",num);
 for(i=1;i<=num;i++)
 {
             scanf("%d",&a[i]);
 }

 merge_sort(1,num);

printf("\nSORTED ORDER: \n");
 for(i=1;i<=num;i++) printf("\t%d",a[i]);
 getch();
 return 0;
}
```

11. SELECTION SORT

```
#include <iostream>
using namespace std;

void swap(int array[], int a, int b) {
  int temp=array[a];
  array[a]=array[b];
  array[b]=temp;
}

void selection_sort(int array[], int size) {
  int i;
  int j;
  int min;
  for(i=0; i<size-1; i++) {
    min=i;
    j=i+1;
    while(j<size) {
      if(array[j]<array[min]) {
        min=j;
      }
      j++;
    }
    swap(array, i, min);
  }
}

void print_array(int array[], int size) {
  int i;
  for(i=0; i<size; i++) {
    cout<<array[i]<<" ";
  }
  cout<<endl;
}

int main() {
  int array[]={5, 50, 4, 78, 2, 100, 40, 500, 450, 32, 210};
  int size=sizeof(array)/sizeof(array[0]);
  print_array(array, size);
  selection_sort(array, size);
  print_array(array, size);
  return 0;
}
```

12. KNAPSACK PROGRAM

```
# 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("n The result vector is:- ");
 for(i=0;i<n;i++)
        printf("%ft",x[i]);

 printf("m Maximum profit is:- %f", tp);

}

void main()
{
 float weight[20], profit[20], capacity;
 int n, i ,j;
 float ratio[20], temp;
 clrscr();

 printf ("n Enter the no. of objects:- ");
 scanf ("%d", &num);

 printf ("n Enter the wts and profits of each object:- ");
 for (i=0; i<n; i++)
 {
 scanf("%f %f", &weight[i], &profit[i]);
 }

 printf ("n enter the capacityacity of knapsack:- ");
 scanf ("%f", &capacity);

 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;
     }
   }
 }

 knapsack(n, weight, profit, capacity);
 getch();
}
```

13. BUCKET SORT

```
#include <stdio.h>

/* Function for bucket sort */
void Bucket_Sort(int array[], int n)
{
    int i, j;
    int count[n];
    for (i = 0; i < n; i++)
        count[i] = 0;

            for (i = 0; i < n; i++)
                (count[array[i]])++;

            for (i = 0, j = 0; i < n; i++)
                for(; count[i] > 0; (count[i])--)
                    array[j++] = i;
        }
        /* End of Bucket_Sort() */

        /* The main() begins */
        int main()
        {
            int array[100], i, num;

            printf("Enter the size of array : ");
            scanf("%d", &num);
```

```c
        printf("Enter the %d elements to be sorted:\n",num);
        for (i = 0; i < num; i++)
            scanf("%d", &array[i]);
        printf("\nThe array of elements before sorting : \n");
        for (i = 0; i < num; i++)
            printf("%d ", array[i]);
        printf("\nThe array of elements after sorting : \n");
        Bucket_Sort(array, num);
        for (i = 0; i < num; i++)
            printf("%d ", array[i]);
        printf("\n");
        return 0;
    }
```