Bubble Sort

```
#include <stdio.h>

#include <conio.h>

void print_array(int arr[],int size) {

 int i;

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

    {

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

    }

    printf("\n");

}

void main() {

    int arr[5],temp,i,j;

    clrscr();

    printf("enter 5 elements : ");
    for(i=0;i<5;i++)

    {

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

    }

    printf("Before Sorting : \n");
    print_array(arr,5);

    for(i=0;i<4;i++)

    {

     for(j=i;j<5;j++)

     {

         if(arr[i]>arr[j])

         {
```

```
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
            printf("after Sorting : \n");
            print_array(arr,5);
          }
      }
    }
    getch();
}
```

Insertion Sort

```
#include <stdio.h>
#include <conio.h>
void print_array(int arr[], int size)
{
   int i;
   for (i = 0; i < size; i++)
   {
      printf("%d , ", arr[i]);
   }
   printf("\n");
}
void main()
{
   int arr[] = {3, 15, 9, 41, 0};
   int i, j, temp, key;
   clrscr();
   printf("show array before sorting:");
   print_array(arr, 5);
   for (i = 1; i < 5; i++)
   {
      key = arr[i];
      j = i - 1;
      for (; j >= 0 && arr[j] > key; j--)
      {
         arr[j + 1] = arr[j];
```

```
        }
        arr[j + 1] = key;
        print_array(arr, 5);
    }
    getch();
}
```

Selection Sort

```
#include <stdio.h>
void print_array(int arr[], int size)
{
   int i;
   for (i = 0; i < size; i++)
   {
      printf("%d , ", arr[i]);
   }
   printf("\n");
}
void main()
{
   int arr[] = {5, 4, 3, 2, 1};
   int i, index = 0, min_val, j;
   clrscr();
   printf("show array before sorting:");
   print_array(arr, 5);
   for (i = 0; i < 4; i++)
   {
      index = i;
      min_val = arr[i];
      for (j = i + 1; j < 5; j++)
      {
         if (min_val > arr[j])
         {
```

```
            index = j;
            min_val = arr[j];
         }
      }
      arr[index] = arr[i];
      arr[i] = min_val;
      print_array(arr, 5);
   }
   getch();
}
```

Merge Sort

```
#include <stdio.h>
#include <conio.h>
/* Function to merge the subarrays of a[] */
void merge(int a[], int beg, int mid, int end)
{
   int n1 = mid - beg + 1;
   int n2 = end - mid;
   int LeftArray[10], RightArray[10]; // temporary arrays
   int i, j, k;
   /* copy data to temp arrays */
   for (i = 0; i < n1; i++)
  {
      LeftArray[i] = a[beg + i];
  }
   for (j = 0; j < n2; j++)
  {
      RightArray[j] = a[mid + 1 + j];
  }
   i = 0;   /* initial index of first sub-array */
   j = 0;   /* initial index of second sub array */
   k = beg; /* initial index of merged sub array */
   while (i < n1 && j < n2)
  {
      if (LeftArray[i] <= RightArray[j])
     {
```

```
            a[k] = LeftArray[i];
            i++;
        }
         else
        {
            a[k] = RightArray[j];
            j++;
        }
         k++;
    }
     while (i < n1)
     {
        a[k] = LeftArray[i];
        i++;
        k++;
     }
     while (j < n2)
     {
        a[k] = RightArray[j];
        j++;
        k++;
     }
}
void mergeSort(int a[], int beg, int end)
{
   if (beg < end)
```

```
   {
      int mid = (beg + end) / 2;
      mergeSort(a, beg, mid);
      mergeSort(a, mid + 1, end);
      merge(a, beg, mid, end);
   }
}
/* Function to print the array */ void printArray(int a[], int
n)
{
   int i;
   for (i = 0; i < n; i++)
   {
      printf("%d ", a[i]);
   }
   printf("\n");
}
void main()
{
   int a[] = {12, 31, 25, 8, 32, 17, 40, 42};
   int n = sizeof(a) / sizeof(a[0]);
   clrscr();
   printf("Before sorting array elements are - \n");
   printArray(a, n);
   mergeSort(a, 0, n - 1);
   printf("After sorting array elements are - \n");
   printArray(a, n);
```

```
    getch();
}
```

Quick Sort

```
#include <stdio.h>
/* function that consider last element as pivot,
place the pivot at its exact position, and place
smaller elements to left of pivot and greater
elements to right of pivot. */
int partition(int a[], int start, int end)
{
   int pivot = a[start]; // pivot element int i = start;
   int j = end;
   int temp;
   while (i <= j)
   {
      while (i <= end && a[i] <= pivot)
      {
         i++;
      }
      while (j >= start && a[j] >= pivot)
      {
         j--;
      }
      if (i <= j)
      {
         temp = a[i];
         a[i] = a[j];
         a[j] = temp;
```

```
      }
   }
   a[start] = a[j];
   a[j] = pivot;
   return j;
}
/* function to print an array */ void printArr(int a[], int n)
{
   int i;
   for (i = 0; i < n; i++)
   {
      printf("%d ", a[i]);
   }
   printf("\n");
}
/* function to implement quick sort */ void quick(int a[], int
start, int end) /* a[] = array to be sorted, start = Starting
index, end = Ending index */
{
   // printArr(a, end-start+1);
   if (start < end)
   {
      int p = partition(a, start, end); // p is the partitioning
index
      quick(a, start, p - 1);
      quick(a, p + 1, end);
   }
```

```
}
void main()
{
   int a[] = {24, 9, 29, 14, 19, 27};
   int n = sizeof(a) / sizeof(a[0]);
   clrscr();
   printf("Before sorting array elements are - \n");
   printArr(a, n);
   quick(a, 0, n - 1);
   printf("\nAfter sorting array elements are - \n");
   printArr(a, n);
   getch();
}
```

Linear Search

```
#include <stdio.h>
#include <conio.h>
int search_element(int arr[], int size, int key)
{
   int i;
   for (i = 0; i < size; i++)
   {
      if (arr[i] == key)
      {
         return i;
      }
   }
   return -1;
}
void main()
{
   int i, key, size, arr[10], index;
   clrscr();
   printf("Enter Array Size : ");
   scanf("%d", &size);
   printf("\n enter array elements : ");
   for (i = 0; i < size; i++)
   {
      scanf("%d", &arr[i]);
   }
```

```
   printf("\n Enter element to be searched : ");
   scanf("%d", &key);
   index = search_element(arr, size, key);
   if (index == -1)
   {
      printf("\n Element is not present in array");
   }
   else
   {
      printf("\n Key element found at %d th index", index);
   }
   getch();
}
```

Binary Search (Iterative)

```
#include <stdio.h>
#include <conio.h>
void print_array(int arr[], int size)
{
   int i;
   for (i = 0; i < size; i++)
   {
      printf("%d , ", arr[i]);
   }
   printf("\n");
}
int binary_search(int arr[], int size, int key)
{
   int mid, low = 0, high = size - 1;
   while (low <= high)
   {
      mid = (low + high) / 2;
      if (arr[mid] == key)
      {
         return mid;
      }
      else if (key > arr[mid])
      {
         low = mid + 1;
      }
      else
      {
         high = mid - 1;
      }
   }
   return -1;
}
void main()
{
   int arr[10], size, key, index, i, j;
   clrscr();
   printf("enter size of array : ");
```

```
   scanf("%d", &size);
   printf("Enter array elements ; ");
   for (i = 0; i < size; i++)
   {
      scanf("%d", &arr[i]);
   }
   printf("Enter Element for searching : ");
   scanf("%d", &key);
   index = binary_search(arr, size, key);
   if (index == -1)
   {
      printf("Element is not present in array");
   }
   else
   {
      printf("Element is present at %d th index", index);
   }
   getch();
}
```

Binary Search (Recursive)

```
#include <stdio.h>
#include <conio.h>
void print_array(int arr[], int size)
{
   int i;
   for (i = 0; i < size; i++)
   {
      printf("%d , ", arr[i]);
   }
   printf("\n");
}
int binary_search(int arr[], int low, int high, int key)
{
   int mid;
   if (low == high)
   {
      if (arr[low] == key)
      {
         return low;
      }
      else
      {
         return -1;
      }
   }
```

```
   else
   {
      mid = (low + high) / 2;
      if (arr[mid] == key)
      {
         return mid;
      }
      else if (key > arr[mid])
      {
         binary_search(arr, mid + 1, high, key);
      }
      else
      {
         binary_search(arr, low, mid - 1, key);
      }
   }
}
void main()
{
   int arr[10], size, key, index, i, j;
   clrscr();
   printf("enter size of array : ");
   scanf("%d", &size);
   printf("Enter array elements ; ");
   for (i = 0; i < size; i++)
   {
```

```
      scanf("%d", &arr[i]);
   }
   printf("Enter Element for searching : ");
   scanf("%d", &key);
   index = binary_search(arr, 0, size, key);
   if (index == -1)
   {
      printf("Element is not present in array");
   }
   else
   {
      printf("Element is present at %d th index", index);
   }
   getch();
}
```

Fractional Knapsack Problem #include <stdio.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);
}
int main()
{
   float weight[20], profit[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], &profit[i]);
   }
   printf("\nEnter the capacityacity of
knapsack:- ");
   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);
   return (0);
}
```

Job Sequence

```
#include <stdio.h>

#define MAX 100

typedef struct Job

{

   char id[5];

   int deadline;

   int profit;

} Job;


void jobSequenceWithDeadline(Job jobs[], int n);

int minValue(int x, int y)

{

   if (x < y)

      return x;

   return y;

}
```

```
int main(void)
{
   // variables
   int i, j;

   // Jobs with deadline and profit
   Job jobs[5] = {
       {"j1", 2, 60},
       {"j2", 1, 100},
       {"j3", 3, 20},
       {"j4", 2, 40},
       {"j5e", 1, 20},
   };

   // temp
   Job temp;
   // number of jobs
   int n = 5;
   // sort the jobs profit wise in descensing order
   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);
   }
   jobSequenceWithDeadline(jobs, n);
   return 0;
}

void jobSequenceWithDeadline(Job jobs[], int n)
{
   // variable
   int i, j, k, maxprofit;
   // free time slots
   int timeslot[MAX];
   // filled Time slots
   int filledTimeSlot = 0;
   // find max deadline value
   int dmax = 0;
```

```
for (i = 0; i < n; i++)
{
   if (jobs[i].deadline > dmax)
   {
      dmax = jobs[i].deadline;
   }
}
// fre time slot initially set to -1
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 all time slot are filled then stop
      if (filledTimeSlot == dmax)
      {
         break;
      }
   }
   // required jobs
   printf("\nRequired Jobs: ");
   for (i = 1; i <= dmax; i++)
   {
      printf("%s", jobs[timeslot[i]].id);
      if (i < dmax)
      {
         printf("--> ");
      }
   }
   // required profit
   maxprofit = 0;

   for (i = 1; i <= dmax; i++)
   {
      maxprofit += jobs[timeslot[i]].profit;
   }
   printf("\n Max Profit: %d\n", maxprofit);
}
```

Prims Algorithm

```
#include <stdio.h>
#include <conio.h>
void prims(int a[10][10], int n)
{
   int
       selected[10],
       t[10][10], sum = 0, min, i, j, x = 0, y = 0, ne = 0;
   for (i = 1; i <= n; i++)
   {
      selected[i] = 0;
   }
   for (i = 1; i <= n; i++)
   {
      for (j = 1; j <= n; j++)
      {
         t[i][j] = 0;
      }
   }
   selected[1] = 1;
   ne = 1;
   while (ne < n)
   {
      min = 1234;
      for (i = 1; i <= n; i++)
      {
```

```
      if (selected[i] == 1)
      {
         for (j = 1; j <= n; j++)
         {
            if (selected[j] == 0 && a[i][j] != 0)
            {
               if (min > a[i][j])
               {
                  min = a[i][j];
                  x = i;
                  y = j;
               }
            }
         }
      }
   }
   t[x][y] = min;
   selected[y] = 1;
   ne = ne + 1;
}
printf("\t\tAns is\n\n");
for (i = 1; i <= n; i++)
{
   for (j = 1; j <= n; j++)
   {
      if (t[i][j] != 0)
```

```
         {
            printf(" (%d,%d)=%d \t\n", i, j, a[i][j]);
            sum = sum + a[i][j];
         }
      }
   }
   printf("\n\ntotal cost = %d", sum);
}
void main()
{
   int i, j, n, edge, a[10][10], first, end, wt;
   clrscr();
   printf("Enter the value of node-> ");
   scanf("%d", &n);
   printf("Enter the value of edge-> ");
   scanf("%d", &edge);
   for (i = 1; i <= n; i++)
   {
      for (j = 1; j <= n; j++)
      {
         // printf("\n\n enter the data for a[%d][%d] =>",i,j);
         // scanf("%d",&a[i][j]);
         a[i][j] = 0;
      }
   }
   for (i = 1; i <= edge; i++)
```

```
   {
      printf("Enter value of starting node-> ");
      scanf("%d", &first);
printf("Enter value of ending
node-> ");
scanf("%d",&end);
printf("Enter Weight of that-> ");
scanf("%d",&wt);
a[first][end]=wt;
a[end][first]=wt;
   }
   printf(" value entered \n\n");
   for (i = 1; i <= n; i++)
   {
      for (j = 1; j <= n; j++)
      {
         printf(" %d\t", a[i][j]);
      }
      printf("\n");
   }
   prims(a, n);
   getch();
}
```

Longest Common Subsequence

```
#include <stdio.h>
#include <string.h>
#define MAX 100 // Define maximum length of input strings
// Function to find LCS and print DP table
void findLCS(char X[MAX], char Y[MAX], int m, int n)
{
   int dp[MAX][MAX];
   // Building the dp table using bottom-up approach
   for (int i = 0; i <= m; i++)
   {
      for (int j = 0; j <= n; j++)
      {
         if (i == 0 || j == 0)
            dp[i][j] = 0;
         else if (X[i - 1] == Y[j - 1])
            dp[i][j] = dp[i - 1][j - 1] + 1;
         else
            dp[i][j] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i -
1][j] : dp[i][j - 1];
      }
   }
   // Print DP Table
   printf("\nLCS DP Table:\n ");
   for (int j = 0; j <= n; j++)
   {
      if (j > 0)
         printf("%c ", Y[j - 1]);
      else
         printf(" ");
   }
   printf("\n");
```

```
for (int i = 0; i <= m; i++)
{
   if (i > 0)
      printf(" %c", X[i - 1]);
   else
      printf(" ");
   for (int j = 0; j <= n; j++)
   {
      printf(" %d", dp[i][j]);
   }
   printf("\n");
}
// Length of LCS
int length = dp[m][n];
printf("\nLength of LCS: %d\n", length);
// Reconstructing LCS
char lcsString[MAX];
int index = length;
lcsString[index] = '\0'; //
Null - terminate the string int i = m, j = n;
while (i > 0 && j > 0)
{
   if (X[i - 1] == Y[j - 1])
   {
      lcsString[index - 1] = X[i - 1];
      i--;
      j--;
      index--;
   }
   else if (dp[i - 1][j] > dp[i][j - 1])
      i--;
   else
      j--;
```

```
   }
   printf("Longest Common Subsequence: %s\n", lcsString);
}
// Main function
int main()
{
   char X[MAX], Y[MAX];
   int m, n;
   // Taking input for two strings  printf("Enter first string: 
");  scanf("%s", X);
   printf("Enter second string: ");
   scanf("%s", Y);
   m = strlen(X);
   n = strlen(Y);
   // Call LCS function  findLCS(X, Y, m, n);
   return 0;
}
```