1. Write a program for the Insertion sort algorithm.

```
#include <stdio.h>
void main()
{
   int n, array[1000], a, b, p;
   printf("Enter number of elements\n");
   scanf("%d", &n);
   printf("Enter %d integers\n", n);
   for (a = 0; a < n; a++)
     scanf("%d", &array[a]);
   for (a = 1 ; a <= n - 1; a++) {
       b = a;
        while (b > 0 && array[b-1] > array[b]) {
            p = array[b];
            array[b] = array[b-1];
            array[b-1] = p;
            b--;
        }
}
printf("Sorted array in ascending order:\n");
for (a = 0; a <= n - 1; a++) {
      printf("%d\n", array[a]);
      }
}
```

2. Write a program for the Selection sort algorithm.

```
#include <stdio.h>
void main()
{
   int array[100], n, a, b, pos, temp;
   printf("Enter number of elements\n");
   scanf("%d", &n);
   printf("Enter %d integers\n", n);
   for (a = 0; a < n; a++)
       scanf("%d", &array[a]);
   for (a = 0; a < (n - 1); a++)
  {
      pos = a;
      for (b = a + 1; b < n; a++)
      {
         if (array[pos] > array[b])
            pos = b;
        }
      if (pos != a)
     {
       temp = array[a];
       array[a] = array[pos];
       array[pos] = temp;
       }
    }
     printf("Sorted array in ascending order:\n");
      for (a = 0; a < n; a++)
         printf("%d\n", array[a]);
      }
```

3. Write a program for Bubble sort algorithm.

```
#include <stdio.h>
void main()
{
   int array[100], n, a, b, temp;
   printf("Enter number of elements\n");
      scanf("%d", &n);
    printf("Enter %d integers\n", n);
    for (a = 0; a < n; a++)
         scanf("%d", &array[a]);
    for (a = 0 ; a < n - 1; a++)
    {
       for (b = 0 ; b < n - a - 1; b++)
        {
```

```
       if (array[b] > array[b+1])
       {
         temp = array[b];
          array[b] = array[b+1];
          array[b+1] = temp;
        }
      }
    }
     printf("Sorted list in ascending order:\n");
     for (a = 0; a < n; a++)
         printf("%d\n", array[a]);
      }
```

4. Write a program for the merge sort algorithm.

```
#include<stdlib.h>
```

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

```
// Merges two subarrays of arr[].

// First subarray is arr[l..m]

// Second subarray is arr[m+1..r]

void merge(int arr[], int l, int m, int r)

{

    int i, j, k;

    int n1 = m - l + 1;

    int n2 =  r - m;
```

```
/* create temp arrays */

int L[n1], R[n2];



/* Copy data to temp arrays L[] and R[] */

for (i = 0; i < n1; i++)

    L[i] = arr[l + i];

for (j = 0; j < n2; j++)

    R[j] = arr[m + 1+ j];
```

```
/* Merge the temp arrays back into arr[l..r]*/

 i = 0; // Initial index of first subarray

 j = 0; // Initial index of second subarray

 k = l; // Initial index of merged subarray

 while (i < n1 && j < n2)

 {

   if (L[i] <= R[j])

   {

     arr[k] = L[i];
```

```
        i++;
    }
    else
    {
        arr[k] = R[j];
        j++;
    }
    k++;
}
```

```
  /* Copy the remaining elements of L[], if there
    are any */
  while (i < n1)
{
     arr[k] = L[i];
     i++;
     k++;
  }
```

```
/* Copy the remaining elements of R[], if there
  are any */
while (j < n2)
{
   arr[k] = R[j];
   j++;
   k++;
}
```

```
}

/* l is for left index and r is right index of the
   sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r)
    {
        // Same as (l+r)/2, but avoids overflow for
```

```
 // large l and h

  int m = l+(r-l)/2;



  // Sort first and second halves

  mergeSort(arr, l, m);

  mergeSort(arr, m+1, r);



  merge(arr, l, m, r);

}
```

```
}

/* UTILITY FUNCTIONS */

/* Function to print an array */

void printArray(int A[], int size)

{

    int i;

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

        printf("%d ", A[i]);
    printf("\n");

}




/* Driver program to test above functions */

int main()

{

    int arr[] = {12, 11, 13, 5, 6, 7};

    int arr_size = sizeof(arr)/sizeof(arr[0]);
    printf("Given array is \n");

    printArray(arr, arr_size);



    mergeSort(arr, 0, arr_size - 1);



    printf("\nSorted array is \n");

    printArray(arr, arr_size);

    return 0;

}
```

5. Write a program for heap sort algorithm.

```
#include<stdio.h>
#include <conio.h>
void Adjust(int Heap_of_Numbers[],int i)  /Function to arrange the elements in the heap/
{
```

```
int j;

int copy;

int Number;

int Reference = 1;

Number=Heap_of_Numbers[0];

while(2*i<=Number && Reference==1)

{

j=2*i;

if(j+1<=Number && Heap_of_Numbers[j+1] > Heap_of_Numbers[j])
```

```
j=j+1;

if( Heap_of_Numbers[j] < Heap_of_Numbers[i])

Reference=0;

else

{

copy=Heap_of_Numbers[i];

Heap_of_Numbers[i]=Heap_of_Numbers[j];

Heap_of_Numbers[j]=copy;

i=j;
}

}

}

void Make_Heap(int heap[])

{

int i;

int Number_of_Elements;

Number_of_Elements=heap[0];

for(i=Number_of_Elements/2;i>=1;i--)
```

```
Adjust(heap,i);

}

int main()

{

int heap[30];

int NumberofElements;

int i;

int LastElement;

int CopyVariable;
```

```
printf("Enter the number of elements present in the unsorted Array:");

scanf("%d",&NumberofElements);

printf("nEnter the members of the array one by one:"); /* Asking for the elements of the unsorted array*/

for(i=1;i<=NumberofElements;i++)

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

heap[0]=NumberofElements;

Make_Heap(heap);

while(heap[0] > 1) /Loop for the Sorting process/

{
```

```
LastElement=heap[0];

CopyVariable=heap[1];

heap[1]=heap[LastElement];

heap[LastElement]=CopyVariable;

heap[0]--;

Adjust(heap,1);

}

printf("nSorted Array:n");/Printing the sorted Array/

for(i=1;i<=NumberofElements;i++)
```

```
printf("%d ",heap[i]);

return 0;

}
```