1.Write a program for the insertion sort algorithm

#include <math.h>

#include <stdio.h>

/\* Function to sort an array using insertion sort\*/

void insertionSort(int arr[], int n)

{

int i, key, j;

for (i = 1; i < n; i++) {

key = arr[i];

j = i - 1;

/\* Move elements of arr[0..i-1], that are

greater than key, to one position ahead

of their current position \*/

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

// A utility function to print an array of size n

void printArray(int arr[], int n)

{

int i;

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

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

printf("\n");

}

/\* Driver program to test insertion sort \*/

int main()

{

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

int n = sizeof(arr) / sizeof(arr[0]);

insertionSort(arr, n);

printArray(arr, n);

return 0;

}

2.Write a program for the selection sort algorithm.

#include <stdio.h>

void swap(int \*xp, int \*yp)

{

    int temp = \*xp;

    \*xp = \*yp;

    \*yp = temp;

}

void selectionSort(int arr[], int n)

{

    int i, j, min\_idx;

    // One by one move boundary of unsorted subarray

    for (i = 0; i < n-1; i++)

    {

        // Find the minimum element in unsorted array

        min\_idx = i;

        for (j = i+1; j < n; j++)

          if (arr[j] < arr[min\_idx])

            min\_idx = j;

        // Swap the found minimum element with the first element

        swap(&arr[min\_idx], &arr[i]);

    }

}

/\* Function to print an array \*/

void printArray(int arr[], int size)

{

    int i;

    for (i=0; i < size; i++)

        printf("%d ", arr[i]);

    printf("\n");

}

// Driver program to test above functions

int main()

{

    int arr[] = {64, 25, 12, 22, 11};

    int n = sizeof(arr)/sizeof(arr[0]);

    selectionSort(arr, n);

    printf("Sorted array: \n");

    printArray(arr, n);

    return 0;

}

3. Write a program for Bubble sort algorithm.

#include <stdio.h>

void swap(int \*xp, int \*yp)

{

    int temp = \*xp;

    \*xp = \*yp;

    \*yp = temp;

}

// A function to implement bubble sort

void bubbleSort(int arr[], int n)

{

   int i, j;

   for (i = 0; i < n-1; i++)

       // Last i elements are already in place

       for (j = 0; j < n-i-1; j++)

           if (arr[j] > arr[j+1])

              swap(&arr[j], &arr[j+1]);

}

/\* Function to print an array \*/

void printArray(int arr[], int size)

{

    int i;

    for (i=0; i < size; i++)

        printf("%d ", arr[i]);

    printf("\n");

}

// Driver program to test above functions

int main()

{

    int arr[] = {64, 34, 25, 12, 22, 11, 90};

    int n = sizeof(arr)/sizeof(arr[0]);

    bubbleSort(arr, n);

    printf("Sorted array: \n");

    printArray(arr, n);

    return 0;

}

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;

}