**I BUBBLE SORT**

SOURCE CODE:

#include <stdio.h>

void bubbleSort(int array[], int size) {

for (int step = 0; step < size - 1; ++step) {

for (int i = 0; i < size - step - 1; ++i) {

if (array[i] > array[i + 1]) {

int temp = array[i];

array[i] = array[i + 1];

array[i + 1] = temp;

}

}

}

}

void printArray(int array[], int size) {

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

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

}

printf("\n");

}

int main() {

int data[] = {-2, 45, 0, 11, -9};

int size = sizeof(data) / sizeof(data[0]);

bubbleSort(data, size);

printf("Sorted Array in Ascending Order:\n");

printArray(data, size);

}

OUTPUT:

Sorted Array in Ascending Order:

-9 -2 0 11 45

**SELECTION SORT**

SOURCE CODE:

#include <stdio.h>

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void selectionSort(int array[], int size) {

for (int step = 0; step < size - 1; step++) {

int min\_idx = step;

for (int i = step + 1; i < size; i++) {

if (array[i] < array[min\_idx])

min\_idx = i;

}

swap(&array[min\_idx], &array[step]);

}

}

void printArray(int array[], int size) {

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

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

}

printf("\n");

}

int main() {

int data[] = {20, 12, 10, 15, 2};

int size = sizeof(data) / sizeof(data[0]);

selectionSort(data, size);

printf("Sorted array in Acsending Order:\n");

printArray(data, size);

}

OUPUT:

Sorted array in Acsending Order:

2 10 12 15 20

**INSERTION SORT**

SOURCE CODE:

#include <stdio.h>

void printArray(int array[], int size) {

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

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

}

printf("\n");

}

void insertionSort(int array[], int size) {

for (int step = 1; step < size; step++) {

int key = array[step];

int j = step - 1;

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

array[j + 1] = array[j];

--j;

}

array[j + 1] = key;

}

}

int main() {

int data[] = {9, 5, 1, 4, 3};

int size = sizeof(data) / sizeof(data[0]);

insertionSort(data, size);

printf("Sorted array in ascending order:\n");

printArray(data, size);

}

OUTPUT:

Sorted array in ascending order:

1 3 4 5 9

**QUICK SORT**

SOURCE CODE:

#include <stdio.h>

void swap(int \*a, int \*b) {

int t = \*a;

\*a = \*b;

\*b = t;

}

int partition(int array[], int low, int high) {

int pivot = array[high];

int i = (low - 1);

for (int j = low; j < high; j++) {

if (array[j] <= pivot) {

i++;

swap(&array[i], &array[j]);

}

}

swap(&array[i + 1], &array[high]);

return (i + 1);

}

void quickSort(int array[], int low, int high) {

if (low < high) {

int pi = partition(array, low, high);

quickSort(array, low, pi - 1);

quickSort(array, pi + 1, high);

}

}

void printArray(int array[], int size) {

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

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

}

printf("\n");

}

int main() {

int data[] = {8, 7, 2, 1, 0, 9, 6};

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

printf("Unsorted Array\n");

printArray(data, n);

quickSort(data, 0, n - 1);

printf("Sorted array in ascending order: \n");

printArray(data, n);

}

OUTPUT:

Unsorted Array

8 7 2 1 0 9 6

Sorted array in ascending order:

0 1 2 6 7 8 9

**MERGE SORT**

SOURCE CODE:

#include <stdio.h>

void merge(int arr[], int p, int q, int r) {

// Create L ← A[p..q] and M ← A[q+1..r]

int n1 = q - p + 1;

int n2 = r - q;

int L[n1], M[n2];

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

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

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

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

// Maintain current index of sub-arrays and main array

int i, j, k;

i = 0;

j = 0;

k = p;

// Until we reach either end of either L or M, pick larger among

// elements L and M and place them in the correct position at A[p..r]

while (i < n1 && j < n2) {

if (L[i] <= M[j]) {

arr[k] = L[i];

i++;

} else {

arr[k] = M[j];

j++;

}

k++;

}

// When we run out of elements in either L or M,

// pick up the remaining elements and put in A[p..r]

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = M[j];

j++;

k++;

}

}

// Divide the array into two subarrays, sort them and merge them

void mergeSort(int arr[], int l, int r) {

if (l < r) {

// m is the point where the array is divided into two subarrays

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

mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

// Merge the sorted subarrays

merge(arr, l, m, r);

}

}

// Print the array

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

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

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

printf("\n");

}

// Driver program

int main() {

int arr[] = {6, 5, 12, 10, 9, 1};

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

mergeSort(arr, 0, size - 1);

printf("Sorted array: \n");

printArray(arr, size);

}

OUTPUT:

Sorted array:

1 5 6 9 10 12