

1. Bubble Sort

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

```
// Function to perform Bubble Sort
```

```
void bubbleSort(int arr[], int n) {  
    for (int i = 0; i < n - 1; i++) {  
        for (int j = 0; j < n - i - 1; j++) {  
            if (arr[j] > arr[j + 1]) {  
                // Swap the elements  
                int temp = arr[j];  
                arr[j] = arr[j + 1];  
                arr[j + 1] = temp;  
            }  
        }  
    }  
}
```

```
// Function to print the array
```

```
void printArray(int arr[], int n) {  
    for (int i = 0; i < n; i++) {  
        printf("%d ", arr[i]);  
    }  
    printf("\n");  
}
```

```
int main() {
```

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

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

```
    printf("Original array:\n");
```

```
    printArray(arr, n);
```

```

        bubbleSort(arr, n);

        printf("Sorted array:\n");
        printArray(arr, n);

        return 0;
    }

2. Selection Sort
#include <iostream>
using namespace std;

void selectionSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
        // Swap the minimum element with the current element
        swap(arr[minIndex], arr[i]);
    }
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

```

```

int main() {
    int arr[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Original array:\n";
    printArray(arr, n);

    selectionSort(arr, n);

    cout << "Sorted array:\n";
    printArray(arr, n);

    return 0;
}

```

3. Insertion Sorting

```

#include <iostream>
using namespace std;

```

```

void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int 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--;
        }
        arr[j + 1] = key;
    }
}

```

```

    }
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

```

```

int main() {
    int arr[] = {12, 11, 13, 5, 6};
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Original array:\n";
    printArray(arr, n);

    insertionSort(arr, n);

    cout << "Sorted array:\n";
    printArray(arr, n);

    return 0;
}

```

4. Merge Sort

```

#include <iostream>
using namespace std;

void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

```

```
int L[n1], R[n2];
```

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

```
    L[i] = arr[left + i];
```

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

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

```
int i = 0, j = 0, k = left;
```

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

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

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

```
        i++;
```

```
    } else {
```

```
        arr[k] = R[j];
```

```
        j++;
```

```
    }
```

```
    k++;
```

```
}
```

```
while (i < n1) {
```

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

```
    i++;
```

```
    k++;
```

```
}
```

```
while (j < n2) {
```

```
    arr[k] = R[j];
```

```
    j++;
```

```
    k++;
```

```

    }
}

void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        merge(arr, left, mid, right);
    }
}

```

```

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

```

```

int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Original array:\n";
    printArray(arr, n);

    mergeSort(arr, 0, n - 1);

    cout << "Sorted array:\n";
}

```

```
    printArray(arr, n);
```

```
    return 0;
```

```
}
```

5. Quick Sort

```
#include <iostream>
```

```
using namespace std;
```

```
int partition(int arr[], int low, int high) {
```

```
    int pivot = arr[high];
```

```
    int i = low - 1;
```

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

```
        if (arr[j] < pivot) {
```

```
            i++;
```

```
            swap(arr[i], arr[j]);
```

```
        }
```

```
    }
```

```
    swap(arr[i + 1], arr[high]);
```

```
    return i + 1;
```

```
}
```

```
void quickSort(int arr[], int low, int high) {
```

```
    if (low < high) {
```

```
        int pi = partition(arr, low, high);
```

```
        quickSort(arr, low, pi - 1);
```

```
        quickSort(arr, pi + 1, high);
```

```
    }
```

```
}
```

```

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

```

```

int main() {
    int arr[] = {10, 7, 8, 9, 1, 5};
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Original array:\n";
    printArray(arr, n);

    quickSort(arr, 0, n - 1);

    cout << "Sorted array:\n";
    printArray(arr, n);

    return 0;
}

```

6. Heap Sort

```

#include <iostream>
using namespace std;

```

```

void heapify(int arr[], int n, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < n && arr[left] > arr[largest])

```



```

        largest = left;

    if (right < n && arr[right] > arr[largest])
        largest = right;

    if (largest != i) {
        swap(arr[i], arr[largest]);
        heapify(arr, n, largest);
    }
}

```

```

void heapSort(int arr[], int n) {
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    for (int i = n - 1; i > 0; i--) {
        swap(arr[0], arr[i]);
        heapify(arr, i, 0);
    }
}

```

```

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

```

```

int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int n = sizeof(arr) / sizeof(arr[0]);
}

```

```
cout << "Original array:\n";  
printArray(arr, n);  
  
heapSort(arr, n);  
  
cout << "Sorted array:\n";  
printArray(arr, n);  
  
return 0;  
}
```