1.Insertion Sort

```cpp
#include <iostream>
using namespace std;
 void insertionSort(int arr[], int n)
{
   int i, key, j;
   for (i = 1; i < n; i++)
   {
       key = arr[i];
       j = i - 1;

       while (j >= 0 && arr[j] > key)
       {
           arr[j + 1] = arr[j];
           j = j - 1;
       }
       arr[j + 1] = key;
   }
}
 void printArray(int arr[], int n)
{
   int i;
   for (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]);
    insertionSort(arr, n);
   printArray(arr, n);
    return 0;
}
```

Output :

```
rs/amzamani/Desktop/sem3/DS/ds lab/assgn09/"insort
5 6 11 12 13
Abus-MacBook-Air:assgn09 amzamani$
```

2. Quick Sort

```
#include <iostream>
using namespace std;
 // A utility function to swap two elements
void swap(int* a, int* b)
{
   int t = *a;
   *a = *b;
   *b = t;
}

int partition (int arr[], int low, int high)
{
   int pivot = arr[high]; // pivot
   int i = (low - 1); // Index of smaller element
    for (int j = low; j <= high - 1; j++)
   {
       // If current element is smaller than the pivot
       if (arr[j] < pivot)
       {
           i++; // increment index of smaller element
           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 size)
{
   int i;
   for (i = 0; i < size; i++)
       cout << arr[i] << " ";
   cout << endl;
}
 // Driver Code
int main()
{
   int arr[] = {10, 7, 8, 9, 1, 5};
   int n = sizeof(arr) / sizeof(arr[0]);
   quickSort(arr, 0, n - 1);
   cout << "Sorted array: \n";
   printArray(arr, n);
   return 0;
}
```

Output

```
Abus-MacBook-Air:assgn09 amzamani$ cd "/Users/amzamani/Desktop/
sem3/DS/ds lab/assgn09/" && g++ quiksort.cpp -o quiksort && "/U
sers/amzamani/Desktop/sem3/DS/ds lab/assgn09/"quiksort
Sorted array:
1 5 7 8 9 10
Abus-MacBook-Air:assgn09 amzamani$
```

3. Merge Sort

```
#include<iostream>
using namespace std;

void merge(int arr[], int l, int m, int r)
{
   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(int i = 0; i < n1; i++)
       L[i] = arr[l + i];
   for(int j = 0; j < n2; j++)
       R[j] = arr[m + 1 + j];

   int i = 0;

   // Initial index of second subarray
   int j = 0;

   // Initial index of merged subarray
   int k = l;

   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 l, int r)
{
    if (l < r)
    {
        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);
    }
}
void printArray(int A[], int size)
{
    for(int i = 0; i < size; i++)
        cout << A[i] << " ";
}
// Driver code
int main()
{
    int arr[] = { 12, 11, 13, 5, 6, 7 };
    int arr_size = sizeof(arr) / sizeof(arr[0]);
    cout << "Given array is \n";
    printArray(arr, arr_size);
    mergeSort(arr, 0, arr_size - 1);
    cout << "\nSorted array is \n";
    printArray(arr, arr_size);
    return 0;
}
```

Output :

```
Abus-MacBook-Air:assgn09 amzamani$ cd "/Users/amzamani/Desktop
/sem3/DS/ds lab/assgn09/" && g++ mergesort.cpp -o mergesort &&
 "/Users/amzamani/Desktop/sem3/DS/ds lab/assgn09/"mergesort
Given array is
12 11 13 5 6 7
Sorted array is
5 6 7 11 12 13 Abus-MacBook-Air:assgn09 amzamani$
```