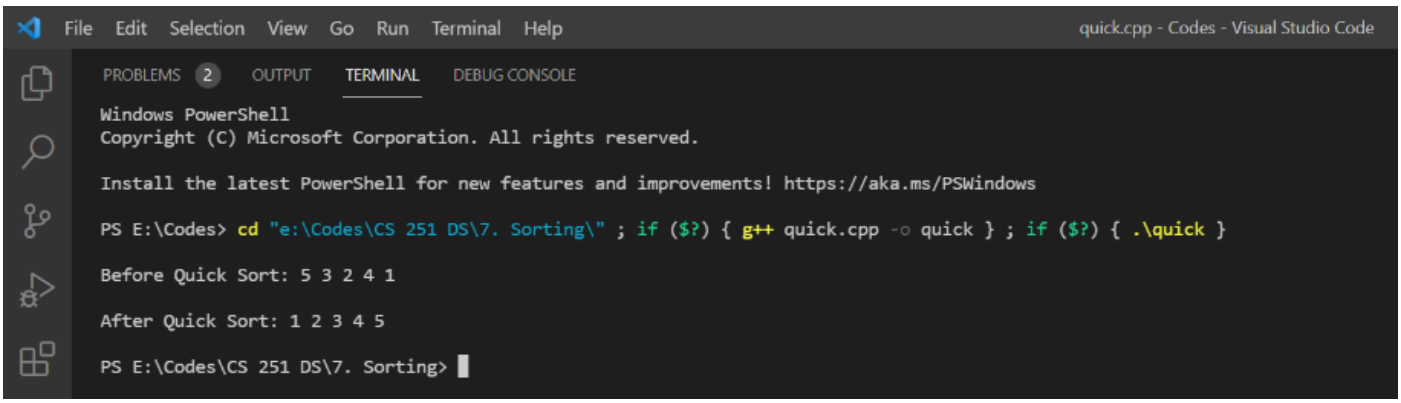
# Quick Sort

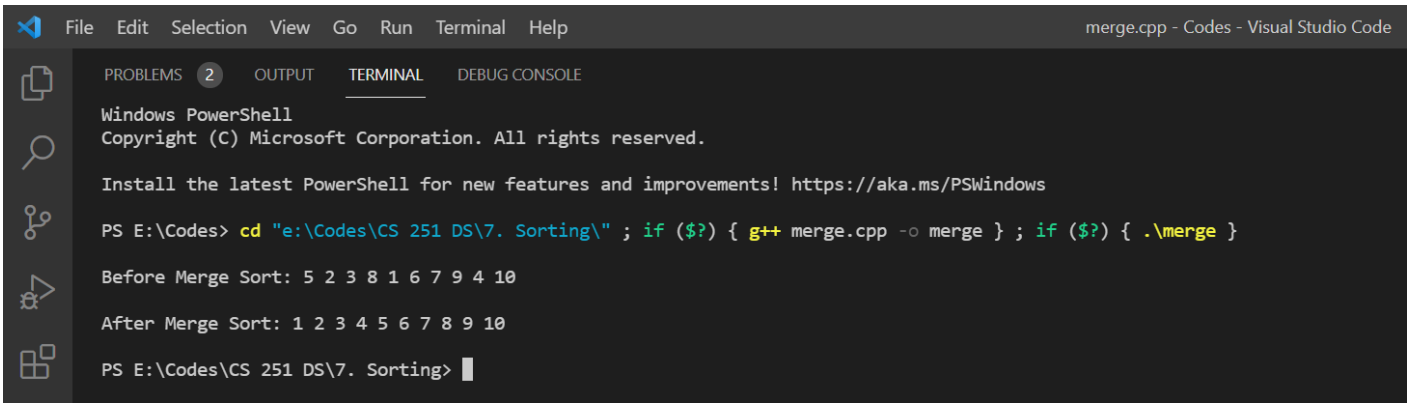
```cpp
// Aneesh Panchal
// 2K20/MC/21

#include<iostream>
using namespace std;

int partition(int a[], int low, int high){
    int pivot = a[high];
    int i = low - 1;
    for (int j = low; j <= high - 1; j++){
        if (a[j] <= pivot){
            i++;
            swap(a[i], a[j]);
        }
    }
    swap(a[i + 1], a[high]);
    return i + 1;
}

void quickSort(int a[], int low, int high){
    int pivot;
    if (low < high){
        pivot = partition(a, low, high);
        quickSort(a, low, pivot - 1);
        quickSort(a, pivot + 1, high);
    }
}

int main(){
    int a[] = {5,4,3,2,1};
    int n = sizeof(a)/sizeof(a[0]);
    quickSort(a, 0, n-1);
    for(int i=0;i<n;i++)
        cout<<a[i]<<" ";
    cout<<endl;
    return 0;
}
```



# Merge Sort

```cpp
// Aneesh Panchal
// 2K20/MC/21

#include<iostream>
using namespace std;

void merge(int a[], int l, int m, int r){
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = a[l + i];
    for (j = 0; j < n2; j++)
        R[j] = a[m + 1 + j];
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2){
        if (L[i] <= R[j]){
            a[k] = L[i];
            i++;
        }
        else{
            a[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1){
        a[k] = L[i];
        i++;
        k++;
    }

    while (j < n2){
        a[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(int a[], int l, int r){
    if (l < r){
        int m = l + (r - l) / 2;
        mergeSort(a, l, m);
        mergeSort(a, m + 1, r);
        merge(a, l, m, r);
    }
}
```

```cpp
int main(){
    int a[] = {5,2,3,8,1,6,7,9,4,10};
    int n = sizeof(a)/sizeof(a[0]);
    mergeSort(a, 0, n-1);
    for(int i=0;i<n;i++){
        cout<<a[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```



# Heap Sort

```cpp
// Aneesh Panchal
// 2K20/MC/21

#include<iostream>
using namespace std;

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

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

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

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

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

    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

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