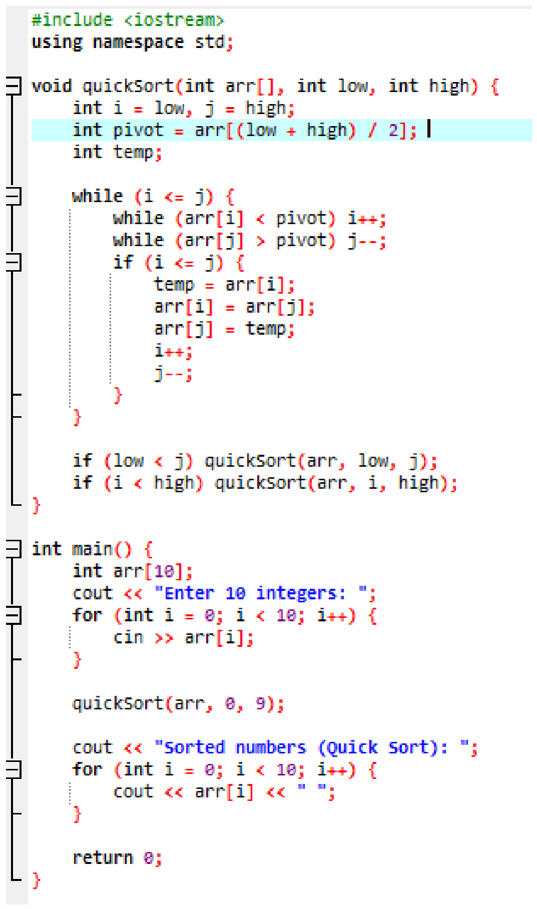
Name:Catungal Kerwin Jan B. **Date: 09/22/25**
Course & Section: CPE 010 - CPE21S4 **Instructor: Engr. Jimlord Quejado**

1. Quick Sort - A quick sort works by picking a pivot element and dividing the list into two groups, smaller and larger than the pivot. It then sorts each group the same way. This keeps repeating until everything is arranged. It is useful for data sets that are large.

2. Shell Sort - Shell sort is like an improved version of insertion sort. Shell sort works on the principle of comparing and sorting based on distance. First, far apart elements are compared and eventually gaps are reduced until they are adjacent, making quick movements happen earlier and allowing for more efficient sorting.

3. Merge Sort - Merge sort works by dividing the list into smaller parts until they each have one element, then merges them in the order they have been arranged.

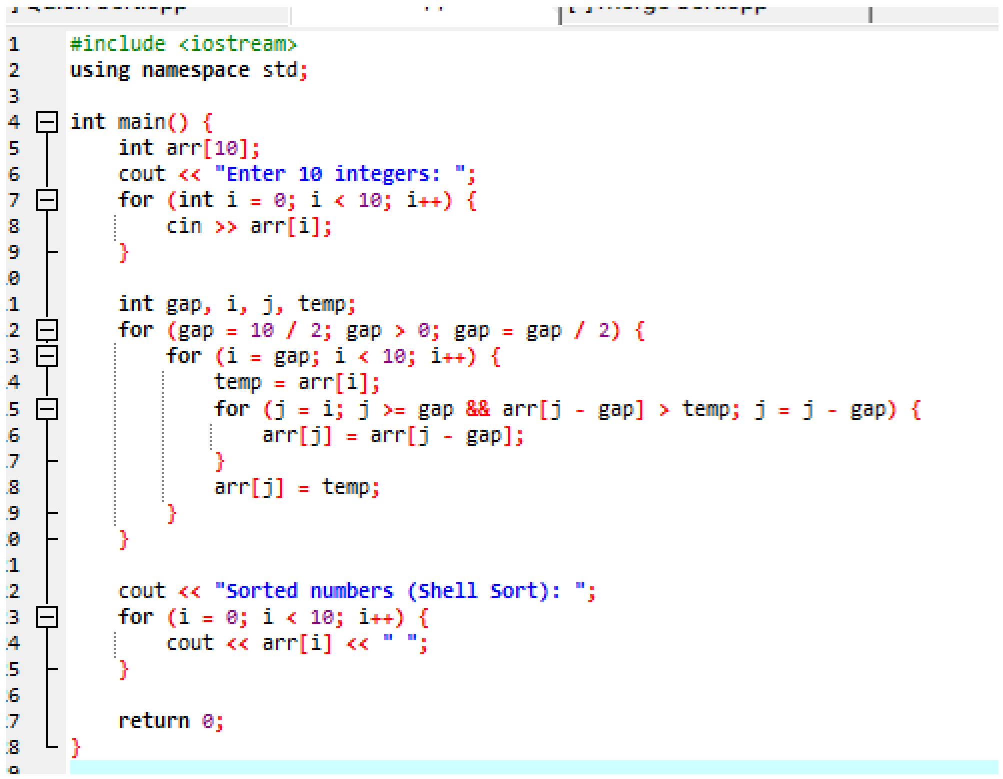
Quick Sort:



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

void quickSort(int arr[], int low, int high) {
    int i = low, j = high;
    int pivot = arr[(low + high) / 2];
    int temp;

    while (i <= j) {
        while (arr[i] < pivot) i++;
        while (arr[j] > pivot) j--;
        if (i <= j) {
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
            i++;
            j--;
        }
    }

    if (low < j) quickSort(arr, low, j);
    if (i < high) quickSort(arr, i, high);
}

int main() {
    int arr[10];
    cout << "Enter 10 integers: ";
    for (int i = 0; i < 10; i++) {
        cin >> arr[i];
    }

    quickSort(arr, 0, 9);

    cout << "Sorted numbers (Quick Sort): ";
    for (int i = 0; i < 10; i++) {
        cout << arr[i] << " ";
    }

    return 0;
}
```

On the quick sort program what i did is for the user to provide 10 digits. Then the program selects a pivot which I used as the middle number and it reorders the array by putting the smaller on the left and the bigger ones on the right. It then recursively divides and sorts each partition until the array is fully sorted. At which point, it displays the sorted digits.

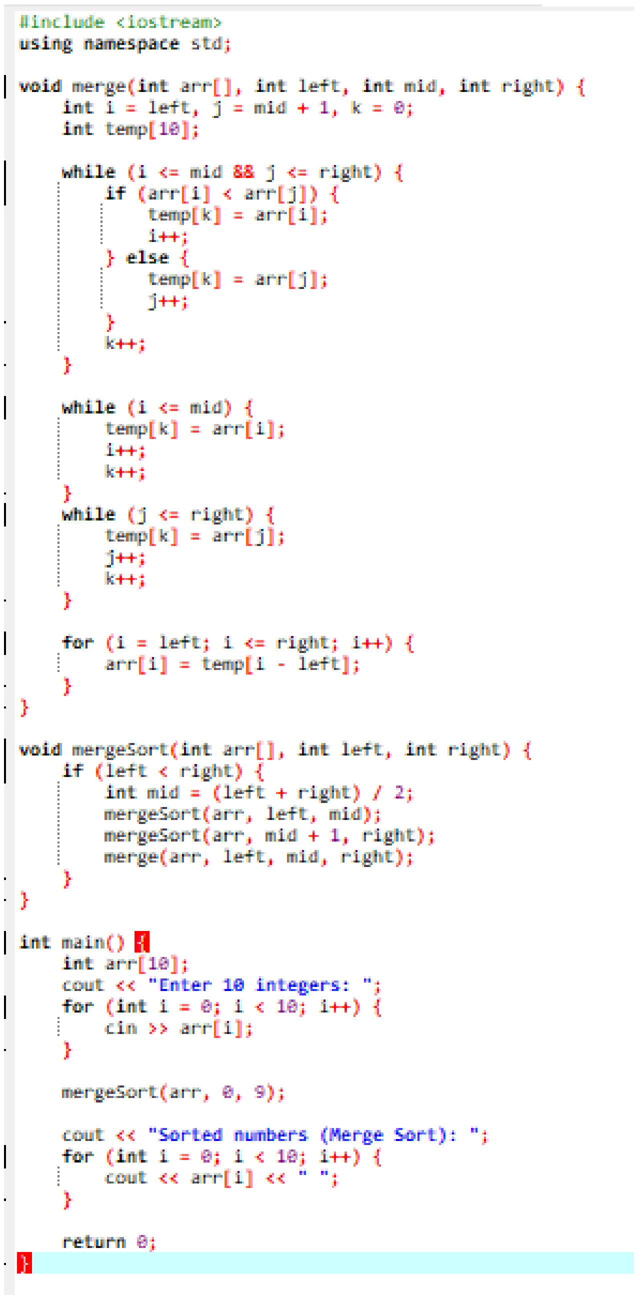
Shell Sort:



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

int main() {
    int arr[10];
    cout << "Enter 10 integers: ";
    for (int i = 0; i < 10; i++) {
        cin >> arr[i];
    }

    int gap, i, j, temp;
    for (gap = 10 / 2; gap > 0; gap = gap / 2) {
        for (i = gap; i < 10; i++) {
            temp = arr[i];
            for (j = i; j >= gap && arr[j - gap] > temp; j = j - gap) {
                arr[j] = arr[j - gap];
            }
            arr[j] = temp;
        }
    }

    cout << "Sorted numbers (Shell Sort): ";
    for (i = 0; i < 10; i++) {
        cout << arr[i] << " ";
    }

    return 0;
}
```

In shell sort, much like the quick sort here also I had the user provide 10 digits. The program then takes a “gap” to start the comparisons of the numbers which is the first step to sorting. The program then iterates and decreases the gap until it reaches a gap of 1 when the algorithm acts like insertion sort. At which point, the list of numbers is sorted in ascending order

Merge Sort:



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

void merge(int arr[], int left, int mid, int right) {
    int i = left, j = mid + 1, k = 0;
    int temp[10];

    while (i <= mid && j <= right) {
        if (arr[i] < arr[j]) {
            temp[k] = arr[i];
            i++;
        } else {
            temp[k] = arr[j];
            j++;
        }
        k++;
    }

    while (i <= mid) {
        temp[k] = arr[i];
        i++;
        k++;
    }
    while (j <= right) {
        temp[k] = arr[j];
        j++;
        k++;
    }

    for (i = left; i <= right; i++) {
        arr[i] = temp[i - left];
    }
}

void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = (left + right) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

int main() {
    int arr[10];
    cout << "Enter 10 integers: ";
    for (int i = 0; i < 10; i++) {
        cin >> arr[i];
    }

    mergeSort(arr, 0, 9);

    cout << "Sorted numbers (Merge Sort): ";
    for (int i = 0; i < 10; i++) {
        cout << arr[i] << " ";
    }

    return 0;
}
```

Here just like the previous two I started with asking the user to provide a list of 10 numbers. The program divides the numbers into smaller units until it cannot be divided anymore. It then combines the units in a certain manner which results in the whole array sorted. The program displays the outcome as the result.

https://www.geeksforgeeks.org/cpp/cpp-program-for-quicksort/
https://www.geeksforgeeks.org/dsa/shell-sort/?adsafe_ip=
https://www.geeksforgeeks.org/dsa/merge-sort/