Name: Vivek Vikram Pundkar

Roll no: 77

GR no: 11910860

Division: C

Batch: 3

Quick Sort using Randomized method

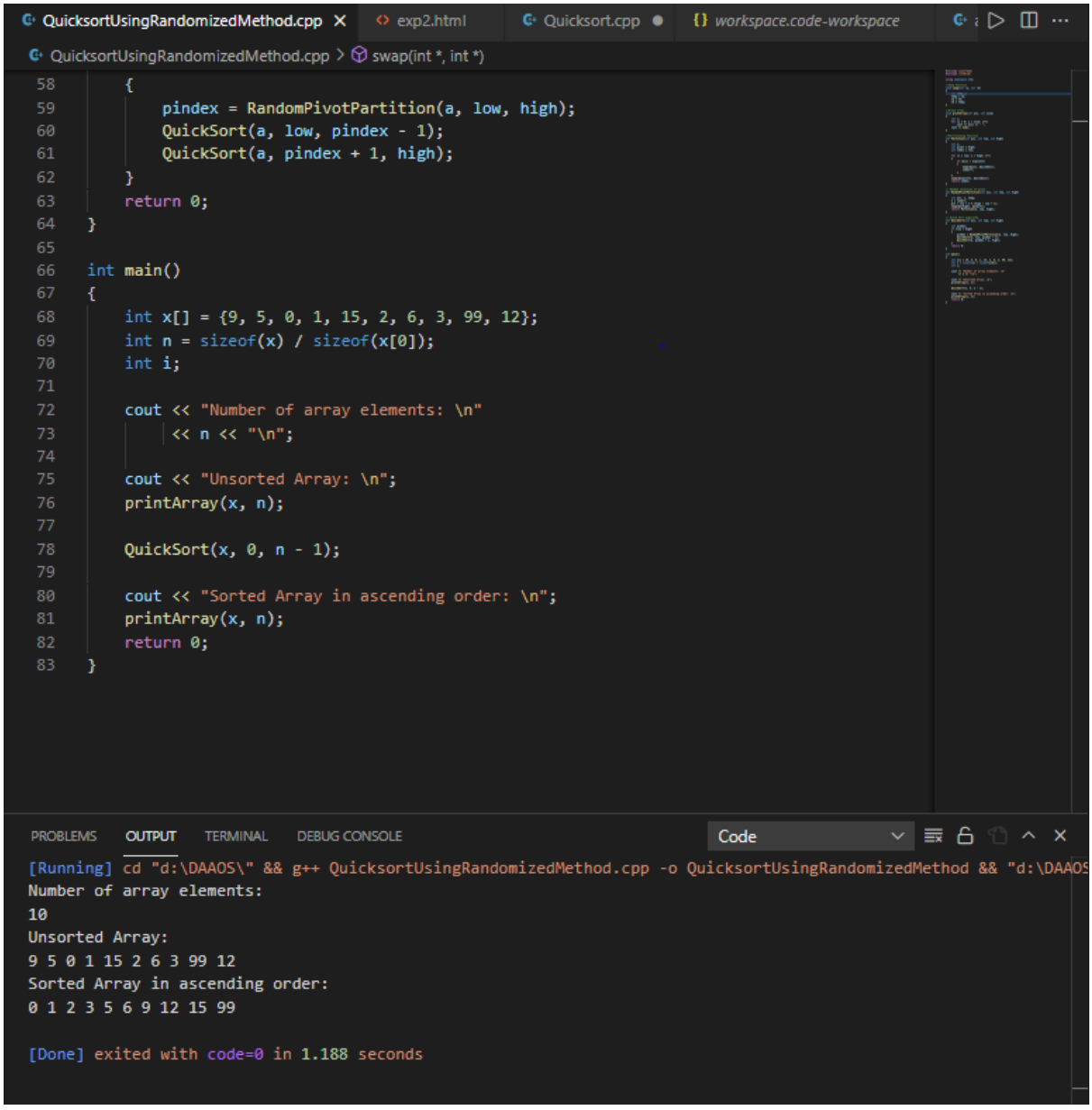
Code:

```
// Random selection of pivot
int RandomPivotPartition(int a[], int low, int high)
{
   int pvt, n, temp;
   n = rand();
   pvt = low + n % (high - low + 1);
   swap(&a[high], &a[pvt]);
   return Partition(a, low, high);
}

// Quick Sort algorithm.
int QuickSort(int a[], int low, int high)
{
   int pindex;
   if (low < high)
   {
      pindex = RandomPivotPartition(a, low, high);
      QuickSort(a, low, pindex - 1);
      QuickSort(a, pindex + 1, high);
```

```
    }
    return 0;
}
```

Output:



Analysis:

Time Complexity of Quick Sort

Best case:

The partition is evenly balanced, here the pivot element is close to middle number or same as middle number. The best-case complexity of the quick sort algorithm is **O(n logn).**

Worst case:

The partition is unbalanced. The worst-case time complexity of Quick Sort is **O(n2).**

Average case:

Here the number of chances to get a pivot element is equal to the number of items. The average case complexity of the quick sort algorithm is **O(n logn).**

Space Complexity of Quick sort

The space complexity is determined on basis of the space used in the recursion stack. The space complexity of quicksort is **O(n*logn).**