Project 1

Aravind Kumar Reddy Yempada
ITCS 6114
800877797

Program Details

The program has been developed in C# using the Visual Studio IDE. Each algorithm has its own class implemented in separate .cs file. The project contains following files:

- LinearSearchProgram.cs
- MergeSortProgram.cs
- HeapSortProgram.cs
- QuickSortProgram.cs
- Program.cs
- CommonClass.cs

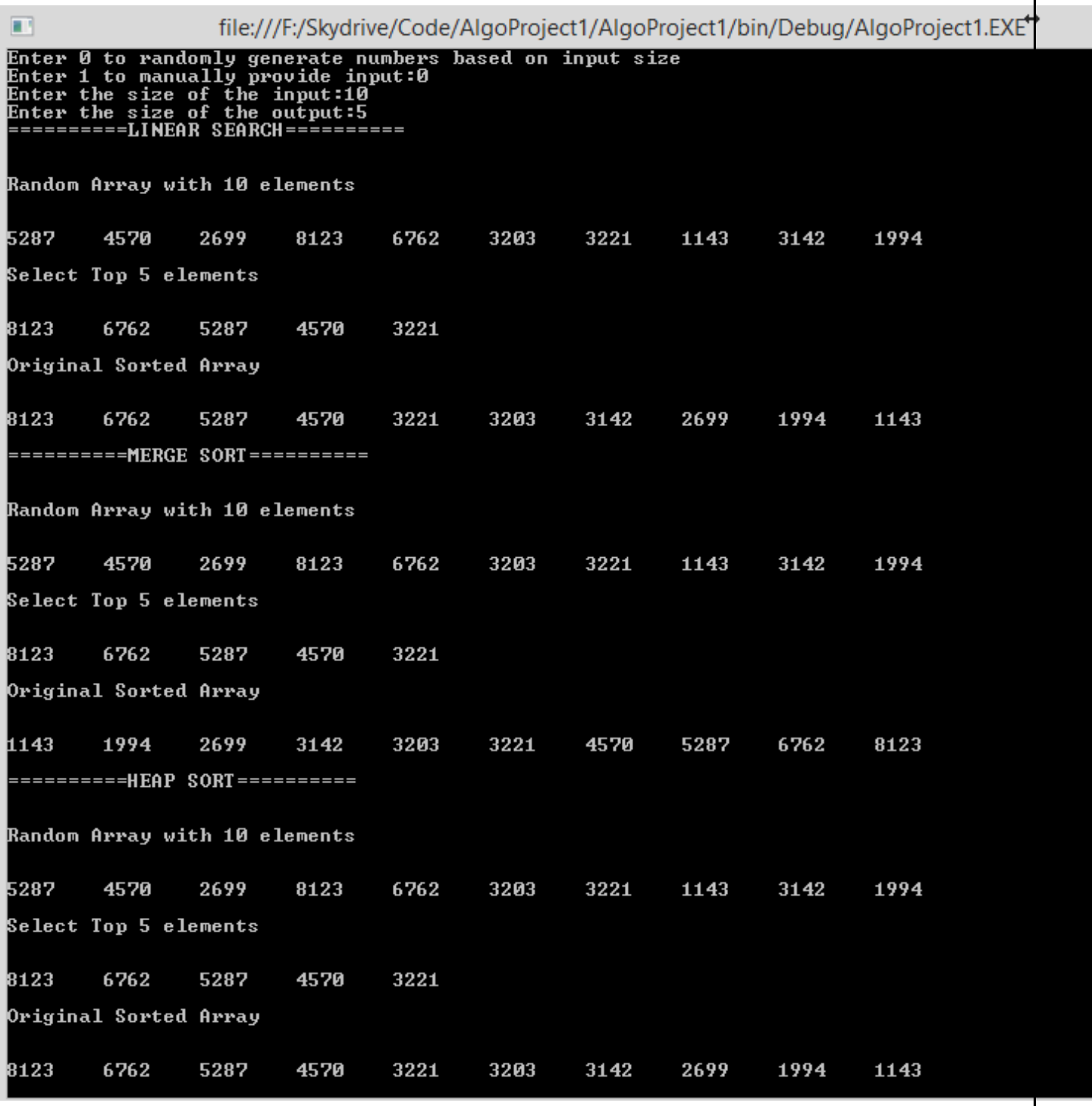
Program.cs contains the Main() method from which the execution starts. The CommonClass.cs contains the methods that can be called by all the classes and it also has the global variable to store the number of operations taken to execute each algorithm.

On execution the program asks for user to select 0 or 1 to generate the input randomly or provide manually respectively. Then the program asks the input size. If 1 is select prior to this, the program takes the input from the user manually .Then it asks for the output size to be displayed. Then there is an option to get detailed output or just the required output. Enter 1 for detailed output (use this only for smaller input) or any other key for minimum output.

The .exe is provided in the Executable folder which can be run to test the program.

Part I

Result Screenshot



```
==========QUICK SORT==========

Random Array with 10 elements

5287    4570    2699    8123    6762    3203    3221    1143    3142    1994
Select Top 5 elements

8123    6762    5287    4570    3221
Original Sorted Array

1143    1994    2699    3142    3203    3221    4570    5287    6762    8123
```

PART II

Result Screenshot

```
Select Top 100 elements

9998 9997 9996 9996 9996 9995 9995 9995 9994 9994 9993 9992 9991 9989 9989 9987 9987 9986 9986 9986
9985 9984 9982 9982 9982 9980 9980 9977 9976 9976 9975 9974 9974 9974 9973 9972 9970 9968 9968 9968
9966 9966 9965 9965 9964 9963 9963 9963 9962 9962 9962 9958 9957 9956 9955 9955 9953 9953 9951 9950
9950 9949 9946 9945 9944 9943 9943 9943 9942 9940 9938 9934 9934 9934 9931 9929 9929 9928 9927 9924
9921 9920 9919 9918 9916 9915 9913 9910 9907 9907 9905 9904 9903 9903 9903 9902 9902 9902 9901 9900

Number of key operations :994950

==========MERGE SORT==========

Select Top 100 elements

9998 9997 9996 9996 9996 9995 9995 9995 9994 9994 9993 9992 9991 9989 9989 9987 9987 9986 9986 9986
9985 9984 9982 9982 9982 9980 9980 9977 9976 9976 9975 9974 9974 9974 9973 9972 9970 9968 9968 9968
9966 9966 9965 9965 9964 9963 9963 9963 9962 9962 9962 9958 9957 9956 9955 9955 9953 9953 9951 9950
9950 9949 9946 9945 9944 9943 9943 9943 9942 9940 9938 9934 9934 9934 9931 9929 9929 9928 9927 9924
9921 9920 9919 9918 9916 9915 9913 9910 9907 9907 9905 9904 9903 9903 9903 9902 9902 9902 9901 9900

Number of key operations :277231
```

```
==========HEAP SORT==========

Select Top 100 elements

9998 9997 9996 9996 9996 9995 9995 9995 9994 9994 9993 9992 9991 9989 9989 9987 9987 9986 9986 9986
9985 9984 9982 9982 9982 9980 9980 9977 9976 9976 9975 9974 9974 9974 9973 9972 9970 9968 9968 9968
9966 9966 9965 9965 9964 9963 9963 9963 9962 9962 9962 9958 9957 9956 9955 9955 9953 9953 9951 9950
9950 9949 9946 9945 9944 9943 9943 9943 9942 9940 9938 9934 9934 9934 9931 9929 9929 9928 9927 9924
9921 9920 9919 9918 9916 9915 9913 9910 9907 9907 9905 9904 9903 9903 9903 9902 9902 9902 9901 9900

Number of key operations :68330

==========QUICK SORT==========

Select Top 100 elements

9998 9997 9996 9996 9996 9995 9995 9995 9994 9994 9993 9992 9991 9989 9989 9987 9987 9986 9986 9986
9985 9984 9982 9982 9982 9980 9980 9977 9976 9976 9975 9974 9974 9974 9973 9972 9970 9968 9968 9968
9966 9966 9965 9965 9964 9963 9963 9963 9962 9962 9962 9958 9957 9956 9955 9955 9953 9953 9951 9950
9950 9949 9946 9945 9944 9943 9943 9943 9942 9940 9938 9934 9934 9934 9931 9929 9929 9928 9927 9924
9921 9920 9919 9918 9916 9915 9913 9910 9907 9907 9905 9904 9903 9903 9903 9902 9902 9902 9901 9900

Number of key operations :165494
```

Empirical Analysis:

LinearSearch : MergeSort : HeapSort : Quicksort = 994950 : 277231 : 68330 : 165494 = 14.5 : 4.05 : 1 : 2.421

PART III:

Time Complexity of Linear search:

To find ith largest element the program executes (n-i) times

$T(n) = n + (n-1) + (n-2) + \ldots (n-i)$

$= n*i - (1+2+\ldots+i)$

$= n*i - C$ where C is some constant

⇨ $T(n) = O(n*i)$

Time Complexity of Merge Sort:

Sorting of n elements (where n>1) using merge algorithm takes time O(nlogn), which is given as

T(n) = 2T(n/2) + Complexity of Merge Function

= 2T(n/2) + n

Using Master Theorem, T(n) = O(nlogn)

It takes constant time to find ith largest element from a sorted array.

Total Complexity is O(nlogn) +i*O(1)

Time Complexity of Heap Sort:

Building a heap from unsorted array takes time O(n)

Deleting a node from a heap takes time O(logn)

For getting i largest elements from a heap we need to make i deletes.

T(n) = O(n) + i*O(logn)

Time Complexity of Quick Sort:

The pivot is selected as the mid element by the selection algorithm. The running time of quick sort will be

T(n) = T(n/2) + n

= O(nlogn)

It takes constant time to find ith largest element from a sorted array.

Total Complexity is O(nlogn)+i*O(1)