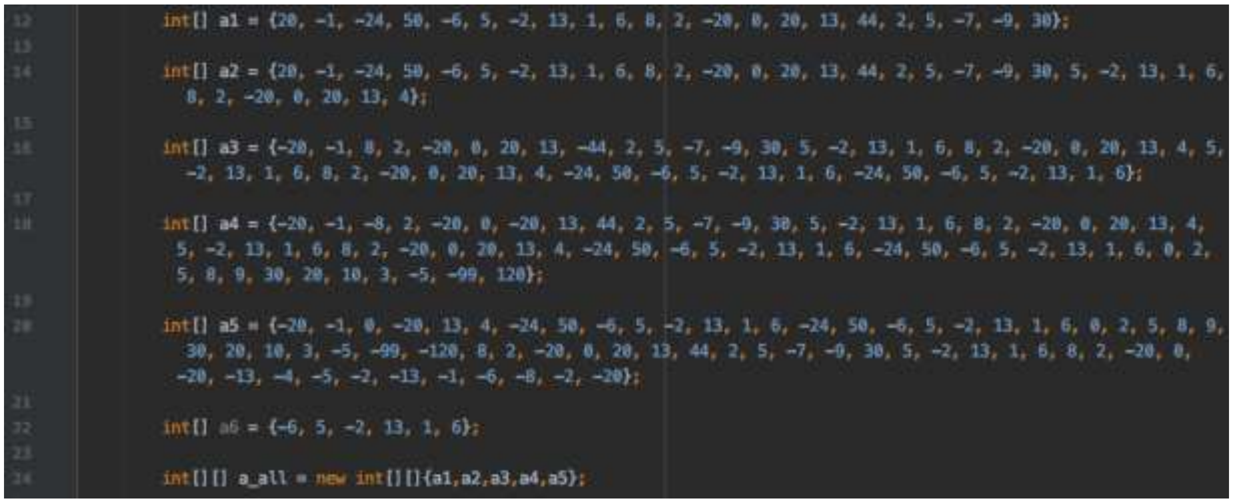
# PROJECT 2

## MAXIMUM SUBARRAY and DIVIDE-CONQUER ALGORITHM

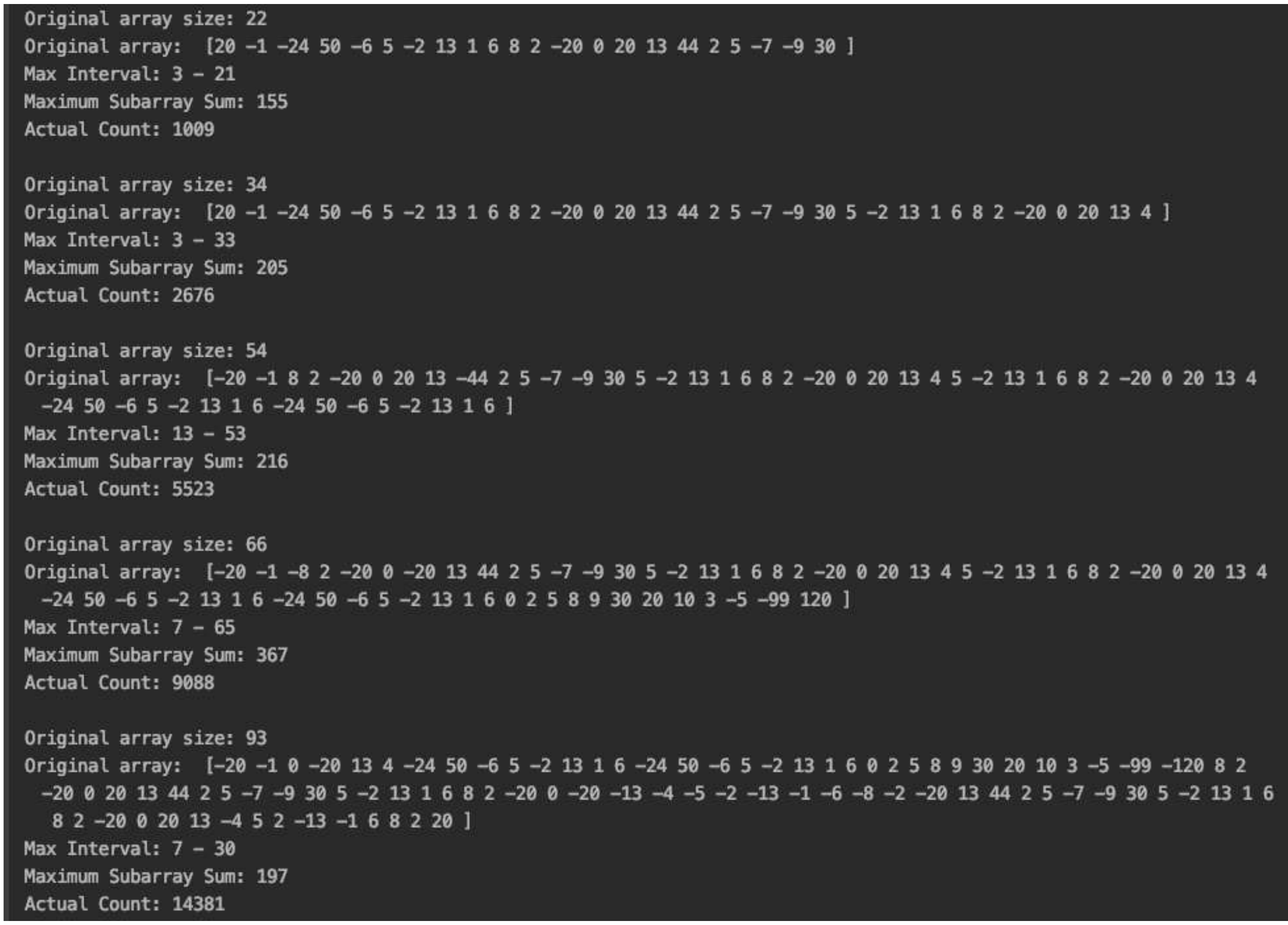
a) Using Maximum contiguous sub-array algorithm Implement a Program in any language you desire (C++ or java) to find a maximum sub-array in a given array of size N,

1. Input at least 5 or more sets of randomized unsorted data with N elements in each set. For example, N= 20, 15, 30, 35, 40, 45, 50. Your array elements must be of real numbers.



```
        int[] a1 = {20, -1, -24, 50, -6, 5, -2, 13, 1, 6, 8, 2, -20, 0, 20, 13, 44, 2, 5, -7, -9, 30};

        int[] a2 = {20, -1, -24, 50, -6, 5, -2, 13, 1, 6, 8, 2, -20, 0, 20, 13, 44, 2, 5, -7, -9, 30, 5, -2, 13, 1, 6,
          8, 2, -20, 0, 20, 13, 4};

        int[] a3 = {-20, -1, 8, 2, -20, 0, 20, 13, -44, 2, 5, -7, -9, 30, 5, -2, 13, 1, 6, 8, 2, -20, 0, 20, 13, 4, 5,
          -2, 13, 1, 6, 8, 2, -20, 0, 20, 13, 4, -24, 50, -6, 5, -2, 13, 1, 6, -24, 50, -6, 5, -2, 13, 1, 6};

        int[] a4 = {-20, -1, -8, 2, -20, 0, -20, 13, 44, 2, 5, -7, -9, 30, 5, -2, 13, 1, 6, 8, 2, -20, 0, 20, 13, 4,
         5, -2, 13, 1, 6, 8, 2, -20, 0, 20, 13, 4, -24, 50, -6, 5, -2, 13, 1, 6, -24, 50, -6, 5, -2, 13, 1, 6, 0, 2,
         5, 8, 9, 30, 20, 10, 3, -5, -99, 120};

        int[] a5 = {-20, -1, 0, -20, 13, 4, -24, 50, -6, 5, -2, 13, 1, 6, -24, 50, -6, 5, -2, 13, 1, 6, 0, 2, 5, 8, 9,
          30, 20, 10, 3, -5, -99, -120, 8, 2, -20, 0, 20, 13, 44, 2, 5, -7, -9, 30, 5, -2, 13, 1, 6, 8, 2, -20, 0,
         -20, -13, -4, -5, -2, -13, -1, -6, -8, -2, -20};

        int[] a6 = {-6, 5, -2, 13, 1, 6};

        int[][] a_all = new int[][]{a1,a2,a3,a4,a5};
```

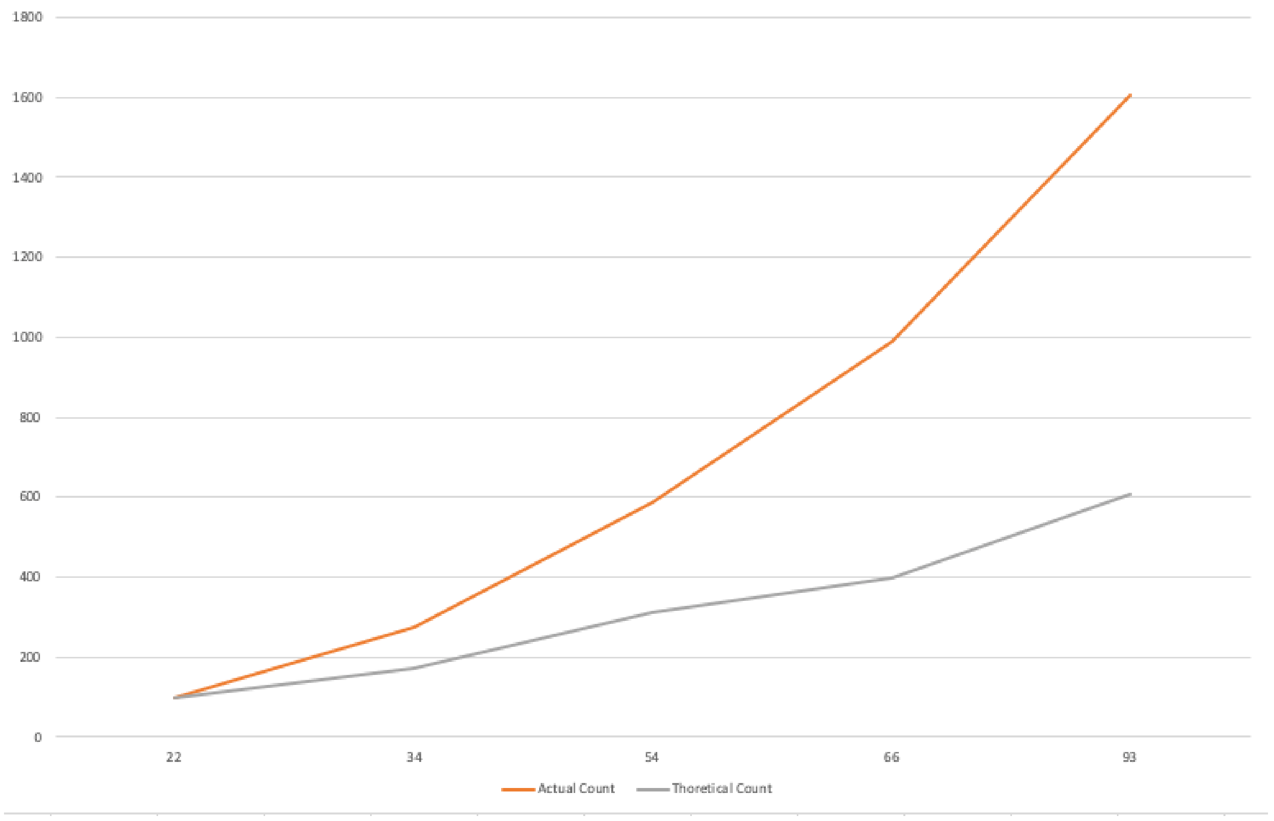
2. Display the original array and the maximum sub-array beginning and ending interval and the sum for each maximum sub-array.



```
Original array size: 22
Original array:  [20 -1 -24 50 -6 5 -2 13 1 6 8 2 -20 0 20 13 44 2 5 -7 -9 30 ]
Max Interval: 3 - 21
Maximum Subarray Sum: 155
Actual Count: 1009

Original array size: 34
Original array:  [20 -1 -24 50 -6 5 -2 13 1 6 8 2 -20 0 20 13 44 2 5 -7 -9 30 5 -2 13 1 6 8 2 -20 0 20 13 4 ]
Max Interval: 3 - 33
Maximum Subarray Sum: 205
Actual Count: 2676

Original array size: 54
Original array:  [-20 -1 8 2 -20 0 20 13 -44 2 5 -7 -9 30 5 -2 13 1 6 8 2 -20 0 20 13 4 5 -2 13 1 6 8 2 -20 0 20 13 4
  -24 50 -6 5 -2 13 1 6 -24 50 -6 5 -2 13 1 6 ]
Max Interval: 13 - 53
Maximum Subarray Sum: 216
Actual Count: 5523

Original array size: 66
Original array:  [-20 -1 -8 2 -20 0 -20 13 44 2 5 -7 -9 30 5 -2 13 1 6 8 2 -20 0 20 13 4 5 -2 13 1 6 8 2 -20 0 20 13 4
  -24 50 -6 5 -2 13 1 6 -24 50 -6 5 -2 13 1 6 0 2 5 8 9 30 20 10 3 -5 -99 120 ]
Max Interval: 7 - 65
Maximum Subarray Sum: 367
Actual Count: 9088

Original array size: 93
Original array:  [-20 -1 0 -20 13 4 -24 50 -6 5 -2 13 1 6 -24 50 -6 5 -2 13 1 6 0 2 5 8 9 30 20 10 3 -5 -99 -120 8 2
  -20 0 20 13 44 2 5 -7 -9 30 5 -2 13 1 6 8 2 -20 0 -20 -13 -4 -5 -2 -13 -1 -6 -8 -2 -20 13 44 2 5 -7 -9 30 5 -2 13 1 6
   8 2 -20 0 20 13 -4 5 2 -13 -1 6 8 2 20 ]
Max Interval: 7 - 30
Maximum Subarray Sum: 197
Actual Count: 14381
```

b) Draw graphs by using counter in the program to compare the actual counting of the algorithm time complexity and the theoretical time complexity.

Obtain two curves for 5-10 sets of randomized unsorted data and put them together in one graph.



| Array Size | Actual Count | Thoretical Count |
|---|---|---|
| 22 | 100 | 98 |
| 34 | 274 | 173 |
| 54 | 588 | 311 |
| 66 | 988 | 399 |
| 93 | 1604 | 608 |

**Notes: Turn in with hard copy and must be professional and the turn in should include the source Program, and the displayed outputs as described above. Proper messages in the output is required to indicate the execution and different outputs. Turn in all assignments in class. It will have late penalty 10% if your turn in 30 minutes after the beginning of the class and the beginning of next class.**