COMP 3330 Analysis of Algorithms
Program#1 – Maximum-Average Subsequence
Due Date: 04/10/17

Overview:

In this assignment you will need to write a C++ program to implement two algorithms for finding the ***maximum-average subsequence*** of length *k* in a sequence of *n* real numbers so that the average of the numbers in the subsequence is maximized. For example, given the sequence <3, 2, 14, 6, 6, 2, 10, 2, 6, 6, 14, 2, 1> and $k = 4$, a subsequence of length *k* with the maximum average value (28/4) is **<2, 6, 6, 14>.** Given the sequence <2, 6.6, 6.6, 3, 7, 6, 7, 2> and $k = 2$, a subsequence of length *k* with the maximum average value (13.2/2) is **<6.6, 6.6>**.

There are several algorithms for solving this problem. You will implement one brute-force algorithm and one divide-and-conquer algorithm to solve this problem. You will also need to analyze the complexity of these algorithms and to conduct experiments to validate your analysis.

For the brute-force approach, you will take a subsequence starting at each position from the beginning of the sequence and compute the average of the values in the subsequence. You will keep track of the maximum average found so far and also the beginning and ending indices of that subsequence.

For the divide and conquer approach, you will divide the input into two halves. The maximum-average subsequence can occur in one of three ways: it can reside entirely in the first half, it can reside entirely in the second half, or it can begin in the first half but end in the second half. You will recursively find the maximum-average subsequence that resides entirely in the first half and the maximum-average subsequence that resides entirely in the second half. You will also find the maximum-average subsequence sum that begins in the first but ends in the second. Then you will choose one of those three subsequences that gives the maximum average.

You will need to develop your program in the Linux environment using the g++ compiler. The program should be executed from the command line:

```
> g++ program1.cpp –o program1
> ./program1 <inputfilename> <k> <-b|-d>
```

inputfilename will be the name of an input file containing the sequence, *k* will be the length of the desired subsequence, and the last option will be used to specify the algorithm (-b for brute-force and –d for divide-and-conquer). The first item in an input file is the value for *n* (the number of real numbers to follow) followed by the numbers in the sequence.

Your program should print the following information to the screen:

- The name of the input file
- The values of *k* and *n*
- The beginning and ending indices of the maximum average subsequence
- The average of the maximum average subsequence
- The time needed for the execution

Experiments:

Experiment#1: Determine the time required for your algorithms as a function of the size of the sequence. Files will be provided containing sequences of length 1000, 5000, 10000, 25000, 50000, 75000, 100000, 500000, and 1000000. Produce a graph that compares the time required by the two algorithms as a function of input size. Use a fixed value for *k* (=50) for these experiments.

Experiment#2: For the longest input file, find the effect of varying the size of *k* (20,40,60,80,100) on the time required by the algorithms. Plot the timing results for both algorithms on one graph. While doing the timing experiments, all I/O should be outside timed portion of the program.

Deliverables:

- You will need to submit the source file ***program1.cpp*** in Blackboard by 04/10/17.
- You will need to turn in a printed report at the beginning of class on 04/10/17 that should contain (a) a title page whose format can be found in Blackboard, (b) a precise pseudocode description (using the notation used in the book) of each algorithm, (c) time complexity analysis of the algorithms—for the brute-force algorithm, setup a summation for the number of basic operation and find the big theta; for the divide-and-conquer, setup a recurrence for the number of basic operation and find the big theta; (d) screenshot of a sample execution of the program, (e) graphs showing the experimental results, and (e) a brief analysis of the results to validate the theoretical time complexities.

Instructions:

- You MUST develop your program in the Linux environment using the g++ compiler. **YOU WILL LOSE 50% OF THE POINTS IF YOUR PROGRAM DOES NOT COMPILE IN LINUX.**
- Use meaningful identifiers, sufficient and helpful comments, and a consistent coding style.
- This will be an individual assignment. However, feel free to contact the instructor (and ONLY the instructor) if you need help. You may discuss general aspects of the assignment with your classmates, but you may not collaborate in any way in producing code. FAILURE TO FOLLOW THESE REQUIREMENTS WILL RESULT IN A GRADE OF F IN THE ASSIGNMENT.

Grading:

This assignment will be graded as follows:

Program Correctness	40%
Program Style	10%
Experiments	20%
Report	30%