CS 158/159 Homework 2

This assignment is worth 15 points and will be **due Monday February 4, 2019 at 11:00pm**. All assignment deadlines are firm and the ability to submit your assignment will be disabled after the deadline elapses. No late work will be accepted. You are encouraged to start this assignment as soon as possible so that you have ample time to seek assistance should you experience difficulties completing or submitting this assignment.

This programming assignment does not have a single solution, and the assignment you submit must be your own original work. **Collaboration with other students is not permitted on homework assignments.** Any submission may be processed with comparison software and the results will be used to detect unacceptable collaboration between individuals. If you need assistance, you should only consult course staff regarding your program.

Your program must adhere to the course programming standards (available in the course packet and in Blackboard). Please review this document before submitting your assignment, because failure to adhere to it will result in a reduction in points. Your program must include the designated program header (~cs15x/student/hdrProg) at the top of the program (which can be inserted in `vi` using the `hp` shortcut while in command mode). The header must include an appropriate description of your program and must contain your official Purdue career account e-mail address. Also note that course standards prohibit the use of advanced programming concepts not yet introduced in the course, unless otherwise specified.

Each of the example executions provided below represents a single execution of the program. Your program must accept input and produce output **exactly** as demonstrated in the example executions. Your program will be tested with the data seen in the examples below and an unknown number of additional tests making use of reasonable data. Do not include any example outputs with your submission.

A single program file (with the `.c` extension) must be submitted electronically via the `guru` server. An example submission was conducted during the first week in lab00. Any attempts to submit via another method will be denied consideration. You may make multiple submissions before the deadline, but only the last attempt is retained and graded. All previous submissions will be over-written and cannot be recovered. The submission script will reject the submission of any file that does not compile. A program must compile to be considered for partial credit. You should always check the confirmation e-mail you receive after a submission to verify that you have submitted the correct file, to the correct assignment, and to the correct lab section. If you have a concern regarding how to submit work, please visit course staff prior to the assignment deadline.

Problem: Data sets are often required to be in a sorted state to make certain types of analysis easier and a variety of sorting algorithms exist to assist scientists and engineers to complete this task. Sorting algorithms are primarily categorized by their run time complexity. This is given in the form of a formula based on the size of the data set (a positive integer `n`). This generates an estimation of how much "work" is necessary to complete the sort. The reason behind the different complexity of sorting algorithms is that some have more (or fewer) instructions ("work") which means that each attempt to sort takes a longer (or shorter) amount of time.

Even among sorting algorithms of the same run time complexity there is another factor related to the time it takes to execute the types of individual instructions used and will be represented by a coefficient by which the formula is multiplied.

Given the data set size, the complexity category, and coefficient for two algorithms determine; (1) the run time for each algorithm, (2) identify the faster algorithm, and (3) the difference in time between the two selections.

Example Execution #1:

```
-=- Algorithm Complexity Options -=-

1. Linear time [n]
2. Logarithmic time [nlog(n)]
3. Exponential time [n^2]

Enter the size of the data set to sort -> 500
Select complexity option #1 -> 1
Enter the coefficient of algorithm #1 -> 10
Select complexity option #2 -> 1
Enter the coefficient of algorithm #2 -> 15

Time for algorithm #1 in seconds: 5000.000
Time for algorithm #2 in seconds: 7500.000
Faster algorithm #: 1
Time difference (seconds): 2500.000
```

Notes on Example Execution #1:

- Both algorithms are from the same complexity category (1).
- The first algorithm has a smaller coefficient than the second.
- The run time is a product of complexity formula, the data set size, and the given coefficient.

Example Execution #2:

```
-=- Algorithm Complexity Options -=-

1. Linear time [n]
2. Logarithmic time [nlog(n)]
3. Exponential time [n^2]

Enter the size of the data set to sort -> 100
Select complexity option #1 -> 1
Enter the coefficient of algorithm #1 -> 125
Select complexity option #2 -> 3
Enter the coefficient of algorithm #2 -> 2

Time for algorithm #1 in seconds: 12500.000
Time for algorithm #2 in seconds: 20000.000
Faster algorithm #: 1
Time difference (seconds): 7500.000
```

Example Execution #3:

```
-=- Algorithm Complexity Options -=-

1. Linear time [n]
2. Logarithmic time [nlog(n)]
3. Exponential time [n^2]

Enter the size of the data set to sort -> 200
Select complexity option #1 -> 2
Enter the coefficient of algorithm #1 -> 5
Select complexity option #2 -> 1
Enter the coefficient of algorithm #2 -> 38.3

Time for algorithm #1 in seconds: 7643.856
Time for algorithm #2 in seconds: 7660.000
Faster algorithm #: 1
Time difference (seconds): 16.144
```

Notes on Example Execution #3:

- Log is base 2.
- `nlog(n)` is the product of `n` and `log(n)`.

Example Execution #4:

```
-=- Algorithm Complexity Options -=-

1. Linear time [n]
2. Logarithmic time [nlog(n)]
3. Exponential time [n^2]

Enter the size of the data set to sort -> 760
Select complexity option #1 -> 1
Enter the coefficient of algorithm #1 -> 760
Select complexity option #2 -> 3
Enter the coefficient of algorithm #2 -> 1

Time for algorithm #1 in seconds: 577600.000
Time for algorithm #2 in seconds: 577600.000
Faster algorithm #: 2
Time difference (seconds): 0.000
```

Notes on Example Execution #4:

- When the time for the two selections is identical then display the second algorithm as being faster.

Additional Notes:

- All floating-point variables must be declared to be of the `double` type to best match our expected results.
- The data set size will always be a positive integer value.
- Complexity options will always be 1, 2, or 3.
- Coefficients will be non-negative.
- This assignment uses logic similar to the selection by calculation examples shown in lecture (p. 83-84 of the course packet).
- Course standards prohibit the use of programming concepts not yet introduced in lecture. For this assignment you can only consider material in the first 3 chapters of the book, notes, and lectures. Use of advanced programming constructs beyond this material would result in a loss of points.