Computer Science 331 — Winter 2016

Assignment #1

Instructions

This assignment concerns lecture material that has been introduced in this course on or before Friday, January 29.

Please **read the entire assignment** before you begin work on any part of it.

This assignment is due by 11:59 pm on Monday, February 8.

Total marks available: 100 (plus 10 bonus marks)

Additional Requirements

- **Start this assignment now!** While the programming required for this assignment is not particularly difficult, the process of performing the analysis that the above questions require will all take thought and time.
- Please make sure to read the information about "How to Succeed in Computer Science 331" as well as "Things to Avoid in Computer Science 331" on the course web site. Note, in particular, the information about academic misconduct that is included there.
- Please read the information about how you should prepare written answers for questions and about how you should submit your material that is provided on the course web page.

Questions

Consider the following computational problem:

Problem: Maximum subsequence sum

Precondition: A is an array (not `null`) of possibly negative integers

Postcondition: S is the maximal value of the sum $\sum_{k=i}^{j} A[k]$ for all possible values of i and j, i.e.,

$$S = \max_{0 \le i,j < n} \sum_{k=i}^{j} A[k] \ .$$

If all entries in A are negative, the maximum subsequence sum is defined to be zero.

An an example, consider the following array:

	0	1	2	3	4	5	6	7	8	9	10	11
A	25	-5	-12	-9	14	12	-13	5	8	-2	18	-8

The values of i and j that maximize the subsequence sum are $i = 4$ and $j = 10$, for which

$$\sum_{k=4}^{10} A_k = 14 + 12 - 13 + 5 + 8 - 2 + 18 = 42 \ .$$

One solution to this problem is to compute the subsequence sum for every possible pair of indices (i, j) and keep track of the maximum value obtained as follows:

```
maxS = 0
n = A.length
for j from 0 to n − 1 do
  for i from 0 to j do
    S = 0
    for k from i to j do
      S = S + A[k]
    end for
    if S > maxS then
      maxS = S
    end if
  end for
end for
```

The file `MaxSubsequenceSum.java` contains an implementation of this algorithm, along with two other methods (one recursive) for solving this problem. You can obtain this file from the course web page. The code in this file is thoroughly documented. It uses exceptions for testing adherence to preconditions for public functions and several assertions in key parts of the code as described in the document "Programming With Assertions" available through the java page on the course web site. You are expected to adhere to this standard of documentation for subsequent assignments in the course.

You should be able to compile and run this program as-is. To run the program with assertion testing enabled, use

`java -enableassertions MaxSubSequenceSum` *command-line args*

This program takes three command-line arguments. Run the program without any to see a description of these required arguments.

1. **(5 marks)** State (no proof required) a loop invariant and loop variant for the outer-most for loop of the algorithm described above.

2. **(20 marks)** Prove that the inner-most loop of the algorithm described above is correct. (Hint: writing pre- and post-conditions specifically for the inner-most loop, as well as re-writing the loop as a while-loop, will be helpful).

3. **(15 marks)** Give an *upper bound* for a function $T_1(n)$ that describes the *worst-case* number of steps executed by the algorithm described above when the input array has n elements. Consider one step to be one assignment, comparison, or arithmetic operation (you may ignore array subscripting). Justify your answer.

4. **(5 marks)** Prove that your function $T_1(n) \in O(n^d)$ for some positive integer d. Your value of d should satisfy $2 \leq d \leq 3$.

5. **(10 marks)** The following pseudocode describes the recursive algorithm used by `maxSubSum2` to solve the maximum subsequence sum problem:

   public int **maxSumRec**(int [] A)
   $n = A.length$
   if $n = 0$ **then**
     return 0
   else if $n = 1$ **then**
     if $A[0] > 0$ **then**
       return $A[0]$
     else
       return 0
     end if
   else
     $S_L =$ **maxSumRec**$(A[0], \ldots, A[n/2])$
     $S_R =$ **maxSumRec**$(A[n/2+1], \ldots, A[n-1])$
     $S_1 = \max(A[n/2], A[n/2] + A[n/2-1], \ldots, A[n/2] + A[n/2-1] + \cdots + A[0])$
     $S_2 = \max(A[n/2+1], A[n/2+1] + A[n/2+2], \ldots, A[n/2+1] + A[n/2+2] + \cdots + A[n-1])$

     return $\max(S_L, S_R, S_1 + S_2)$
   end if

   Give a recurrence relation $T_2(n)$ that describes the *worst-case* number of steps executed by this algorithm when the input array has n elements. Consider one step to be one assignment, comparison, or arithmetic operation (you may ignore array subscripting). You may express functions involved in the statement of the recurrence relation using asymptotic notation, for example, use $\Theta(n)$ in place of any linear functions of n that arise.

6. **(10 marks)** Using mathematical induction, it is possible to prove that the correct recurrence $T_2(n) \in \Theta(n \log n)$. It is also easy to see that $T_3(n) \in \Theta(n)$, where $T_3(n)$ describes the worst-case running time of function `maxSubSum3`. Using asymptotic notation as described in the lectures, relate $T_1(n)$ to $T_2(n)$, $T_1(n)$ to $T_3(n)$ and $T_2(n)$ to $T_3(n)$. You do not need to justify these answers — just state the *most precise* relationship that is appropriate.

7. **(10 marks)** Describe a set of black-box tests for the maximum subsequence sum problem based on the problem description given at the beginning of the assignment description.

8. **(10 marks)** Describe a set of white-box tests for the function `maxSubSum3`.

9. **(10 marks)** Implement a JUnit test harness called `MSSTester.java` that applies your black box test cases from Problem 7 for the MaxSubSequenceSum problem to all three implementations (`maxSubSum1`, `maxSubSum1`, `maxSubSum1`) and your white-box test cases from Problem 8

to the `maxSubSum3` function. Be sure to include this file with your submission of the assignment.

10. **(5 marks)** Your tests should have uncovered (at least) two errors, one in each of the `maxSubSum2` and `maxSubSum3` functions. Describe these errors and correct them in the `MaxSubsequenceSum.java` file. Be sure to include the corrected version with your submission of the assignment.

Bonus Questions

11. **(10 marks)** Prove that the function `maxSubSum2` runs in worst-case time $O(n \log n)$.