Assignment 2

David Blumenthal, Florian Hofer, Werner Nutt, Daniele Porello

Recursion and Algorithms for Arrays

Instructions: Your assignment should represent your own effort. However, you are not expected to work alone. It is fine to discuss the exercises and try to find solutions together, but each student shall write down and submit his/her solutions separately. It is good academic standard to acknowledge collaborators, so if you worked together with other students, please list their names.
You can write up your answers by hand (provided your handwriting is legible) or use a word processing system like Latex or Word. Experience shows that Word is in general difficult to use for this kind of task.
For all programming tasks, it is not allowed to use any external libraries ("import") if not stated otherwise.
Please, include name, student ID, code of your lab group (A, B, or C) and email address in your submission.

JUnit tests: For this assignment, we have designed tests that your code should satisfy. We did this using the JUnit framework for Java. (You will learn a lot more about this in the course "Advanced Programming" by Barbara Russo.) To get started with your coursework, you just need to know two things:

- For each question, we have already declared a class and a method. You just have to write code to implement the method.
- To implement the methods, use the Eclipse development environment, to which you have been introduced in your introductory programming course.

To set up Eclipse and get started, follow the steps below:

1. Download the zip file `DSA_A2.zip` to some directory and unzip it. (That zip file contains a "project".)
2. Import the project into Eclipse, that is, open the menu item *File* → *Import* ... and choose *Existing Projects into Workspace* from the Import dialog.

At this point, you can consult the Package Explorer and you will find the classes under the *src* node and the tests under the *test* node. Implement your methods by refining the ones you find in the classes. Make sure that they pass the JUnit tests in the test directory. In the next lab, you will learn how to use the JUnit framework in Eclipse.

1. Recursion

A palindrome is a phrase that reads the same forward and backward (examples: ‘racecar’, ‘radar’, ‘noon’, or ‘rats live on no evil star’). By extension we call every string a palindrome that reads the same from left to right and from right to left.
Develop a recursive algorithm that takes as input a string and decides whether the string is a palindrome.

1. Write down your algorithm in pseudocode.
2. Implement your algorithm in the `PalindromeChecker` class.

(6 Points)

2. Maximal Length of Ascents in Arrays

Consider an array $A[1..n]$ of integers. A *subarray of A* is a contiguous segment of A. We denote the subarray from position k to position l as $A[k..l]$.
The subarray $A[k..l]$ is an *ascent* if $A[j] \leq A[j+1]$ for all j where $k \leq j < l$. In other words, an ascent is a nondecreasing segment of A.
We want to compute the maximal length of an ascent in A. For instance, for the array $A = [3, 1, 4, 2, 4, 4, 5, 3]$, the maximal length of an ascent would be 4, because the subarray $A[4..7] = [2, 4, 4, 5]$ is the longest ascent in that array.
We are interested in an *efficient* algorithm.

1. Write down pseudocode for an iterative algorithm `maxAscent`(A) that takes an array A of integers as input and returns the maximal length of an ascent in A.
2. Explain why your algorithm is correct.
3. Implement your algorithm in the `MaxAscentsInArrays` class.

(10 Points)

3. Array of Averages

Design an *efficient* algorithm that achieves the following task: Given an array $A[1..n]$ of floating point numbers, it returns a two-dimensional array, say M, of size $n \times n$ in which the entry $M[i][j]$ for $i \leq j$ contains the *average* of the array entries $A[i]$ through $A[j]$. That is: if $i \leq j$, then

$$M[i][j] = \frac{A[i] + \cdots + A[j]}{j - i + 1},$$

whereas for $i > j$ we have that $M[i][j] = 0$.

1. Describe your idea for an algorithm that creates this matrix.
2. Write down the algorithm in pseudocode.
3. How many assignments will your algorithm perform for an input of size n?
4. Implement the algorithm in the `ArrayOfAverages` class.

5. Write code to measure the running time of your Java program for random inputs of size $n = 10$, 100, 1000, etc. Does the growth of the running time correspond to your estimate of the number of assignments?

(14 Points)

Deliverables.

1. For each question, hand in the Java file that you wrote.
2. Write one report for all tasks.

Therefore, you need to submit four Java files (three for the classes to implement and one for the measurements for question 3) and one report. Please combine all deliverables into one zip file, which you submit via the OLE website of the course.

Submission: Until Tue, 21 March 20176, 23:55 hrs, to the OLE submission page of

Lab A / Lab B / Lab C