Assignment 6

**David Blumenthal, Florian Hofer,
Werner Nutt, Daniele Porello**

Matching Pairs and Sorting

1. Matching Pairs

Let s be an integer and A an array of integers. Then A has a *matching pair* for s if there are two distinct positions i, j such that $A[i] + A[j] = s$. The matching pair problem is to check, given s, whether A has a matching pair for s. Your goal is to develop an *efficient* algorithm that solves the matching pair problem.

1. Write pseudocode for an efficient algorithm `hasMatchingPair`.
   Hint: An algorithm that you know from the lecture may be useful.

2. What is the asymptotic worst-case complexity of your algorithm? Explain your answer.

3. Explain why your algorithm is correct. That is, show that whenever your algorithm returns `true`, there are two values in A that add up to s, and that if your algorithm returns `false`, there are no such values.
   Hint: Choose the right loop invariant.

4. Implement your algorithm in the class `MatchingPair`, which you find in the zip file `DSA_A6.zip`.

(16 Points)

2. Person Sorter

In this assignment we are applying sorting algorithms to objects instead of numbers. In the `DSA_A6` zip file, you find templates for the three classess related to this assignment, that is, the classes

1. `Person`,

2. `PersonSorter`, and

3. `PersonSorterTest`, which contains the JUnit tests.

Your task is to complete the class `PersonSorter`. The `PersonSorter` class provides methods to ascendingly sort an array of instances of `Person` by a field contained in the class `Person`. We provide you with two constants for two sorting modes: `BY_LAST_NAME` and `BY_DATE_OF_BIRTH`, each of which corresponds to a field of the class `Person`. The method `sortBy` then takes an array of `Person` and a specified sorting mode as inputs and returns the sorted array. For the class `PersonSorter`, your task is as follows.

1. Implement the two sorting methods mentioned above. Use a different sorting algorithm for each method.

2. Implement as well a method `sortByLastNameAndDateOfBirth`. This method is to sort an array of `Person` object by the last name and the date of birth in lexicographic order, which means the following: Consider p_1 and p_2 to be two elements of the input array. If p_1 and p_2 have different last names, then arrange them by their last name. Otherwise, if they have the same last name, then arrange them by their date of birth. Use also a different sorting algorithm for this method.

Finally, test your sorting algorithm using the JUnit tests provided in the class `PersonSorterTest`.

(14 Points)

Deliverables.

1. Your implementation of the class `MatchingPair` in Exercise 1 and the class `PersonSorter` in Exercise 2.

2. Write one report for all tasks.

Combine all deliverables into one zip file, which you submit via the OLE website of the course. Please, follow the "Instructions for Submitting Course Work" on the Web page with the assignments, when preparing your coursework.
Submission: Until Mon, 2 May 2016, 23:55 hrs, to the OLE submission page of

Lab A / Lab B / Lab C