14 credits

26956(b) Software Workshop (MSc) Spring Term

Submit your answers to the sub-questions as separate files in a single **zip** file. Your answers should be written as Java or PDF files as appropriate.

(a) **Testing**

Consider a method for finding the *unique* elements among two lists that are sorted in strictly ascending order. That is, the output list will contain exactly those elements that appear in one of the input lists but not the other. The output list will be sorted in strictly ascending order. Input **a** does not contain duplicates, and neither does input **b**.

```
static List<Integer> unique(List<Integer> a, List<Integer> b)
```

Decide 10 independent test cases for the **a**, **b** combinations, and in a table, state for each test case (i) what property of the method you are trying to test, (ii) the inputs, and (iii) the expected result or effect. You may write lists using the square bracket notation, e.g., [1, 2, 3]. Try to make your test cases small. Make sure that the "boundary cases" are tested. **[10 marks]**

(b) **Recursion**

Using recursion, write an efficient implementation of the method **unique** specified in part (a). You may use the **List** class from Term 2. **[8 marks]**

(c) **Collections**

You are asked to design and implement a solution for a lecturer for managing team projects for their class. A **Student** class is already available, which stores, for each student, the ID number, last name and first name. It has a **toString** method and implements the **Comparable** interface for comparing students by their last name-first name combination. The class also has a field for storing the *team name* for each student, which is a string unique to the team.

You need to define a class **Teams** in which the information pertaining to each team can be stored. It should store at a minimum the students who are members of the team. It should be extensible so that, in future, other information can be added such as the team supervisor, meeting schedules, marks etc. Frequent operations include updating information about individual teams, and tabulating all teams in an alphabetical order.

Describe what data structure you would use to store the teams and discuss its efficiency considerations. **[7 marks]**

Define the **Teams** class along with

- a constructor that takes an array of **Student** objects.
- methods to *add* and *delete* students.
- a **print** method that prints all the teams along with their members in the format

  team : member 1, member 2, ...

  where the teams as well as members appear in sorted order. **[15 marks]**

You will be assessed for the quality of your solution in addition to its correctness.