TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica

TU Delft

Exam IN3205 / IN3420
Software Quality and Testing

4 April 2008

- There are 5 questions worth of 23 points in total.
- Total number of pages: 2.
- Use of books and readers is not allowed
- Write clearly and avoid verbose explanations: Points may be deducted for unclear or sloppy answers
- You can answer in English or in Dutch
- Please list your answers in the right order
- The tentative grading scheme is:

Question:	1	2	3	4	5	Total
Points:	3	3	6	6	5	23

- If p is the number of points you score, the exam grade E will most likely be determined by

$$E = 1 + 9 * p/23$$

- Your final grade F is determined by the average of your labwork L and the exam E:

$$F = (E + L)/2$$

  Note that you can only pass if both $E \geq 5.5$ and $L \geq 5.5$.

- If you want to receive the grade under the old code (IN3420, IN4088), indicate that code explicitly on your form.

GOOD LUCK!

1. (3 points) While working for software company C, you created a test and integration plan for software system X. Key steps in your plan include ensuring (as close as you can get to) 100% statement coverage for the components used or developed, and a period of 4 months intensive integration testing.

   Your manager M studies the plans, and proposes to reduce the period of integration testing to just a single month, based on the argument that with 100% statement coverage there is no need for so much integration testing.

   Write a memo of at most 150 words to your manager, in which you include at least three good reasons in defense of your original plan.

2. During his guest lecture, Ivo de Jong from ASML explained the test and integration strategy in use at ASML.
   (a) (1 point) During the lecture, three key performance indicators related to testing were discussed. What were they, and which one is given top priority at ASML?
   (b) (1 point) Describe the implications of optimizing this indicator on the test strategy at ASML.
   (c) (1 point) Is this strategy also suitable for testing aeroplane software, for example at a company like Airbus? Explain your answer.

```
/**
  * Determine if the other cell is an immediate
  * neighbour of the current cell.
  * @return true iff the other cell is immediately adjacent.
  */
public boolean adjacent(Cell otherCell) {
   boolean result;

   boolean xdiffersone = Math.abs(getX()-otherCell.getX()) == 1;
   boolean ydiffersone = Math.abs(getY()-otherCell.getY()) == 1;

   if ((xdiffersone && !ydiffersone) || (!xdiffersone && ydiffersone)) {
       result = true;
   } else {
       result = false;
   }
   return result;
}
```

Figure 1: Implementation of adjacent

3. While implementing the "adjacent" method in class Cell from JPacman, your partner proposes the solution from Figure 1. The intent of the method is that it returns true if the cell is immediately next to `otherCell`, and false in all other cases.

   While studying the implementation, you quickly discover a an incorrect case (at distance greater than one), which was not covered by the two test cases provided by your partner. You therefore deicde to analyze whether there is a test adequacy criterion with which your partner should have found the bug.

   (a) (1 point) Construct the control flow graph of the "adjacent" implementation, taking Java's short-circuit evaluation into account.

   (b) (1 point) How many paths from start to end does this graph contain?

   (c) (1 point) How many paths are needed at least to achieve statement coverage?

   (d) (1 point) How many paths are needed at least to achieve branch coverage?

   (e) (1 point) Are any of the paths in the graph infeasible? If so, which ones?

   (f) (1 point) Does adhering to either statement or branch coverage necessarily lead to the detection of the fault? Explain your answer.

4. Next, you decide to analyze the intended behavior of the adjacent method from Figure 1:

   (a) (1 point) Create a decision table in which you indicate what the result of the adjacent method should be. As basic conditions, disinguish the $(2 * 3 = 6)$ cases that the absolute difference between the x (respectively y) coordinate of the current and the other cell equals zero, equals one, and is greater than one. (You may want to give names $C_1, C_2, \ldots$ to the columns, and $R_1, R_2, \ldots$ to the rows, to make it easier to refer to them later on.)

   (b) (1 point) Are there any constraints on permitted combintations of the basic conditions? If so, which ones?

   (c) (1 point) How many test cases are needed at least to achieve *basic condition coverage*? Explain your answer.

   (d) (1 point) How many test cases are needed at least to achieve *compound condition coverage*? Explain your answer.

   (e) (2 points) How many test cases are needed at least to achieve Modified Condition/Decision Coverage (MC/DC)? Explain your answer.

5. Next, you decide to try to rethink the robustness of the `adjacent` method.

   (a) (1 point) Identify assumptions the `adjacent` method relies on.

   (b) (1 point) Provide a design of the interface of this method that adheres as much as possible to the principles of defensive programming.

   (c) (1 point) Provide a design that adheres as much as possible to the principles of design by contract.

   (d) (1 point) Let class A be a subclass of class B, and assume that this subclass relation adheres to the principles of design by contract. Furthermore, let m be a method in B that is overridden in A. Which of the method implementations, $A.m$ and $B.m$, is the most defensive? Explain your answer.

   (e) (1 point) Briefly compare the main risks and benefits of defensive programming versus design-by-contract.

End of exam