The exam is open-book and open-laptop. Web browsing is allowed, but you are not allowed to use e-mail clients nor Instant Messaging clients, nor to share any information “live” with anybody inside or outside the exam room.

At the end of the exam you must submit both the question sheets and your answer sheets. To avoid that any of your solutions get lost, make sure to write your full name (and student ID) on each sheet of paper that you submit. Also, please number the pages on your answer sheets.

Write clearly. Answers that are illegible cannot be counted as correct answers. Only answers written in English will be marked.

Total marks: 40 (equivalent to 40% total grade). You must get at least 10 marks in this exam to not fail the course.

PART 1: Multiple-Choice Questionnaire (10 marks)
PART 2: Open Questions & Constructive Tasks (30 marks)

Total: 40 marks (=100%)
PART 1: Multiple-Choice Questionnaire (10 marks)

Important: For Part 1, please check boxes on the separate questionnaire answer sheet. Read carefully before you answer and observe instructions carefully!

The following questions (up to question Q-08) have exactly one correct answer, thus, you must check exactly one answer box on the separate answer sheet. If you think that more than one answer is correct, choose the one answer that seems to be most correct/suitable/relevant.

Q-01 (1 mark): Which of the following activities is a validation activity?
Answer choice:
A: Code inspection (done by testers and developers)
B: Regression testing
C: Unit testing
D: Requirements inspection (done by customer)

Q-02 (1 mark): Which of the following statements describes a fault?
Answer choice:
A: An expression in Java code uses the operator ‘+’, while using ‘++’ would have been correct
B: A programmer misunderstood an interface specification
C: A program crashes when a tester enters a valid input
D: A reviewer reports that he/she found a defect in a requirements spec. Later it turns out to be a ‘false positive’

Q-03 (1 mark): Which of the following statements about the relationships between control-flow coverage criteria is correct?
Answer choice:
A: Condition Coverage subsumes Statement Coverage
B: Decision Coverage subsumes Statement Coverage
C: Decision Coverage subsumes Condition Coverage
D: Condition Coverage subsumes Condition/Decision Coverage

Q-04 (1 mark): Which of the following statements describes ‘Selective Regression Testing’ best?
Answer choice:
A: After any kind of change, all existing code will be re-tested
B: Only after a bug fix, all code will be re-tested
C: Only code affected by a code change will be re-tested
D: Only after a functional enhancement, all code will be re-tested

Q-05 (1 mark): Which statement about Data-Driven Testing (DDT) is not correct?
Answer choice:
A: DDT is the execution of the same set of test steps with multiple (different) data
B: In DDT, test data (i.e., inputs and expected outcomes) are integrated with the test scripts
C: In DDT, test data (i.e., inputs and expected outcomes) for all test cases are read from an external data source
D: In DDT it is easy to maintain test data and test scripts by different roles (e.g., customer representatives and professional testers, respectively)
Q-06 (1 mark): Which of the following statements about code refactoring and testing is correct?  
Answer choice:  
A: Before refactoring a code module, a test suite for this code module must be in place  
B: Refactoring is done in order to add and test new functionality  
C: Refactoring is a test activity  
D: Refactoring does neither influence (or change) program design nor existing test cases

Q-07 (1 mark): Which of the following statements about test techniques is correct?  
Answer choice:  
A: Mutation testing does require a test oracle derived from the program specification  
B: Equivalence class partitioning is a white-box testing technique  
C: White-box testing techniques exploit knowledge about the code that is tested  
D: Black-box testing techniques exploit knowledge about the code that is tested

Q-08 (1 mark): You execute a program and you see the following message on the screen: 'Program aborted due to division by 0'. Which of the following terms describes your experience best?  
Answer choice:  
A: I made an error (while using the program)  
B: I debugged an error  
C: I localized a fault  
D: I triggered a failure

The following question (question Q-09) can have more than one correct answer. You must check all correct answer choices to get full marks. You get partial marks, if you check some of the correct answer choices. You will get a penalty, if you check an incorrect answer choice. You don’t get a penalty, if you miss a correct answer choice. Overall, the lowest possible mark you can get is 0 (i.e., even if everything you check is wrong, you won’t get a negative mark).

Q-09 (2 marks): You plan to test the correct functioning of a text editor’s 4 font effects: bold, italics, underline, and strikethrough. Each effect can be on or off. Which statements are correct?  
Answer choice:  
A: Complete testing of all possible combinations of font effects requires 16 test cases  
B: The number of pairwise interactions between font effects is 16  
C: Complete pairwise testing of font effects requires not more than 8 test cases  
D: Complete testing of all 4-way interactions of font effects requires more than 16 test cases
PART 2: Open Questions & Constructive Tasks (30 marks)

Task 1 (6 marks):

The figure below shows an example of a system with a mismatch between its specified and implemented functionality.

![Diagram of system with mismatch]

a) Briefly describe the essentials of white-box and black-box testing. In addition, give one example of a white-box and one example of a black-box testing strategy.

*Hint: Test levels are not examples of testing strategies.*

b) Say for each set of functionality, (i), (ii) and (iii), whether it may be tested by white-box and/or black-box testing strategies. Justify/explain your answer briefly.

Task 2 (5 marks):

An simple editor may have four input parameters with values as shown below. Note that one parameter has 3 values while the other three parameters have only 2 values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font type</td>
<td>Times</td>
<td>Arial</td>
<td>Calibri</td>
</tr>
<tr>
<td>Font style</td>
<td>normal</td>
<td>bold</td>
<td></td>
</tr>
<tr>
<td>Font size</td>
<td>large</td>
<td>small</td>
<td></td>
</tr>
<tr>
<td>Font colour</td>
<td>black</td>
<td>red</td>
<td></td>
</tr>
</tbody>
</table>

a) If you wanted to test all possible combinations of parameter values (i.e., the number of all 4-way interactions), how many test cases would you need? Show your calculation.

b) How many pairwise interactions (i.e., 2-way interactions) between parameter values exist? Show your calculation.

c) Show that you can cover all 2-way interactions between parameter values with only 6 test cases. Present the set of test cases (per test case you need to specify only the values of the input parameters)
Task 3 (6 marks):

Consider a software module that is intended to accept the name of a grocery item, \textit{gname}, and a list of the different sizes the item comes in, \textit{gsize1} to \textit{gsize5}, specified in litres. A maximum of five sizes may be entered for each item.

The specification states that the item name is to be entered first, followed by a comma, then followed by a list of sizes. A comma must be used to separate each size. Spaces (blanks) are to be ignored by the software anywhere in the input. The input must be entered in one line. The item name is to be alphabetic characters with a length of 2 to 15 characters. Each size may take a value in the range of 1 to 10, whole numbers (integers) only. The sizes are to be entered in ascending order (smaller sizes first).

Based on this specification, many Equivalence Classes for testing the software module can be derived. The following list shows some examples:

1. \textit{gname} is alphabetic (valid)
2. \textit{gname} is not alphabetic (invalid)
3. \textit{gsizeX} value is less than 1 (invalid)
4. \textit{gsizeX} value is in the range 1 to 10 (valid)
5. ...

Add 6 additional equivalence classes (comprising at least 2 invalid input classes) and provide a set of test cases that cover the 10 classes listed (i.e., the 4 classes above and your 6 classes). Try to make the set of test cases as small as possible. Remember that a complete test case also contains an output value. To make things easy, assume that there are only two output values, i.e., ‘input accepted’ and ‘invalid input’, for valid and invalid inputs, respectively.

\textit{Hint: First list the equivalence classes then list the test cases. Say for each test case which equivalent class(es) have been covered.}

(6 marks)

Task 4 (10 marks):

For the method \textit{speedingfine} below, perform tasks a), b) and c):

```java
public static int speedingfine (int age, int overspeed; int licencemark) {
    int fine = 0;
    if ((age >= 25) && (overspeed < 30) && (licencemark < 3))
        fine = fine + 100 * overspeed;
    else {
        if ((age < 25) || (licencemark >= 3))
            fine = fine + (200 * overspeed);
        if (overspeed >= 30)
            fine = fine + 5000;
    }
    return fine;
}
```

(10 marks)
a) Draw the control flow graph (CFG) and calculate the McCabe Cyclomatic number, i.e., the number of linearly independent paths.

*Hint:*
*Use the line numbers of the code example above to label the nodes in your CFG.*  
(2 marks)

b) Write down a minimal set of test cases needed to achieve 100% statement coverage. For each test case, state the path in the CFG that it covers.  
(4 marks)

c) 100% Decision coverage requires that all decisions are evaluated at least once to ‘true’ and once to ‘false’. Write down the test cases needed to achieve 100% Decision coverage. The total number of test cases should be minimal. Show how each test case satisfies the Decision coverage criterion, i.e., say for each test case how each decision is evaluated.  
(4 marks)

*Hint:*
*Remember that complete test cases include both input values and expected output values.*

Task 5 (3 marks):

Three reviewers have inspected a document and found the defects shown in Table 1 below. Use a capture-recapture model to estimate the number of remaining defects in the document. You must show the details of your calculation to get marks.

<table>
<thead>
<tr>
<th>Defect</th>
<th>Reviewer 1</th>
<th>Reviewer 2</th>
<th>Reviewer 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D4</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D5</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>D6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D7</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D8</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D9</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>D10</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D11</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D12</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D13</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D14</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>D15</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D16</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(3 marks)