1 Introduction
Recall that in black box testing we tested the functionality of the application. In contrast, in white-box testing we will test the software using information about the internal structure of the application. White box testing can be applied at the unit, integration, and system level testing. There are different coverage criterion depending on the level of detail of the white-box testing.

2 Learning Objectives
This exercise aims at giving an understanding of white-box testing. The specific learning goal is to gain an insight into the effectiveness of white-box testing using different coverage criterion.

3 Preparation (Individual Work)
3.1 Background information on necessary topics on white box testing can be found in [1] and in the lecture slides on White-box test techniques.
3.2 Look at the pseudocode below and draw its flowgraph on paper. Write test cases that gives you 100% statement coverage. Show your work to the lab supervisor. This will not be used for marking.

**** Pseudocode ****
Start
input (score);
if (score < 45) then
print ("Fail");
else
print ("pass");
if (score > 80) then
print ("with distinction");
End
4 Exercise (In Pairs)
In this lab session you will test the NextDate.java program with white-box testing techniques. Get an understanding of how the program works, the code is available at the course web-site. Make flowgraphs of the methods in the NextDate class.

Get familiarized with the Eclipse IDE [5], the JUnit unit testing framework [2] [3], and the Java code coverage tool EclEmma [4].

Prepare test cases based on the flowgraph. Implement the test cases using different coverage criterion.

5 Report
The report should be submitted in a PDF file via the course web-page selecting Lab 3. Details about the content of the report are given in the marking section below.

On the first page of your report, write the name of the lab (White Box Testing), names of ALL group members, university IDs of ALL group members, and email addresses of group members.

6 Marking
Record your test results carefully in your report. The report must include answers to the following questions:

1) Draw the flowgraph of method run() and any one other method in NextDate.java program. (0.75 + 0.25 point)
2) Prepare an initial set of TEN test cases aimed at achieving as high Statement Coverage (SC) as possible. Note that the EclEmma tool interpret "Statement" as an instruction in the code. Report the test cases using the test case format in Appendix 1. (2 points)
3) Execute these test cases, and report the result in the defects report. (0.5 point)
4) Report with EclEmma screenshot what percentage of total instructions were covered. In order to get full points for this section you must cover greater-than-zero instruction coverage of all functions. (0.5 point)
5) Next, check with same set of test cases what percentage of branches are covered and report it with EclEmma screenshot. (0.5 point)
6) Next, add at least FIVE test cases aimed at increasing Branch Coverage (BC), and report them in the test case format in Appendix 1. Also at this stage you must detect at least FOUR issues in the NextDate.java program. Less than FOUR issues will subtract 0.5 point. (1 point)
7) Next, execute the newly created test cases and report the result in the defects report. (0.5 point)
8) Report with EclEmma screenshot whether BC increased or not after adding the newly created test cases. (0.5 point)
9) Report with EclEmma screenshot whether SC increased or not after adding the newly created test cases. (0.5 point)
10) Which coverage criterion works best and why? (0.5 point)
11)Compare black-box testing and white-box testing. Discuss advantages and disadvantages with each and when they are appropriate to use? (0.5 point)
12) Provide constructive feedback on the lab, which will help us to improve the lab next year. (1 point)
13) Actively working on writing test cases during lab and showing at least **FOUR** test cases to the lab instructor. (1 point)

References

**Appendix 1: Sample Test Case (Example test case, not to refer to this code)**

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Method Under Test</th>
<th>Input</th>
<th>Expe. Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>isTwentyEightDayMonth()</code></td>
<td>2</td>
<td>True</td>
</tr>
</tbody>
</table>

**Appendix 2: Sample Defects Report**

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Method Under Test</th>
<th>Description of Defect</th>
<th>Expected Result</th>
<th>Actual Result</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>isTwentyEightDayMonth()</code></td>
<td>When input is '2' output obtained is 'False', where the actual output should be 'True'.</td>
<td>True</td>
<td>False</td>
<td>Severe defect</td>
</tr>
</tbody>
</table>