Instructions

- Submission deadline: Lab reports must be submitted within seven days. For example, if your lab takes place on a Tuesday, then you have to submit your report no later than on the following Monday, 23:59 hours.
- Late submission policy:
  - 50% of the total marks deducted for submission up to 24 hours late
  - 100% of the total marks deducted for submission more than 24 hours late
- Group: There should be a maximum of two members in a group. Answers should be your own group work, explained in your own words. If you work in a group of two students, make sure to mention the names of both students in the submitted lab report.
- Maximum amount of points is nine (9).

1 Introduction
In black-box testing, the purpose is to test the different functionalities of an application without looking into the internal structure of the program. There are different strategies to use in order to test efficiently. In this exercise, you will use equivalence class partitioning (ECP) and boundary-value analysis (BVA) on a small application, where the main function takes three positive whole-number triangle side lengths to be typed in as command-line arguments, and the program responds with a description of the triangle.

2 Learning Objectives
The exercise aims at giving an understanding of black-box testing. The specific learning goal is to gain an insight into two common black-box testing methods: equivalence class partitioning and boundary-value analysis.

3 Preparation (Individual Work)
3.1 Background information on necessary topics can be found in the lecture slides on Black-box testing methods.
3.2 Read the documentation for the program Triangle, available at the course homepage.

4 Homework (Group Work)
Use the jar file named lab2.jar to test the triangle application using equivalence class partitioning and boundary-value analysis. Specify test cases by using the ECP and BVA
methods at a unit test level. Remember to specify test input, test description, and expected output, and make sure that the test cases cover both valid equivalence classes and invalid equivalence classes.

To proceed with the task, first, prepare a list of equivalence classes (EC) on paper. Then derive a list of test cases based on listed ECs and their boundaries. An example of the test case format is given in Appendix 1. Then execute the test cases. Add new test cases if you discover some equivalence classes you missed during the preparation.

Guidelines for finding ECs:
1. The tester must consider both valid and invalid equivalence classes. Invalid classes represent erroneous or unexpected inputs.
2. Equivalence classes may also be defined for output conditions.
3. The derivation of input and output equivalence classes is a heuristic process. The conditions that are described in the lecture slides only give the tester tips for identifying the partitions. There is no unique set of partitions. Given the same set of conditions, individual testers may make different choices of equivalence classes.
4. In some cases, it is difficult for the tester to identify equivalence classes. The conditions/boundaries that help define classes may be absent, or obscure, or there may seem to be a very large or very small number of equivalence classes for the problem domain. These difficulties may arise from an ambiguous, contradictory, incorrect, or incomplete specification and/or requirements description. It is the duty of the tester to seek out the analysts and meet with them to clarify these documents. Additional contact with the user/client group may be required.

5 Reporting
The report should be submitted in a PDF file via the course wiki page selecting HW2.

On the first page of your report, write the name of the lab (Black Box Testing), names of ALL group members, university IDs of ALL group members, and email addresses of group members. The report is expected to be about 4 to 5 A4 size pages.

6 Content of Report and Marking
Record your test results carefully in your test report. Remember to specify test case ID, what is tested, input, expected output, and any other useful information. You may also want to make room for any other relevant information such as the severity of the defects found. The report must include the following:

1. List of equivalence classes. We expect to see at least 15 ECs. Remember that equivalence classes have to be defined for each input and output field of the program. (3 points)
2. List of test cases that will cover all equivalence classes (listed under point 1) and also take into account boundaries. You must define at least 20 test cases, cover all ECs and trigger at least five failures to get full marks. See appendix-1 for the expected test case reporting format. Remember to also add boundary values (BV) to some of your test cases. (5 points)
3. Execute the test cases and report the outcome in the failure report. See appendix-2 for the expected failure reporting format. (1 point)
### Part 1 - Appendix 1: Sample Test Case

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Functionality Under Test</th>
<th>Input</th>
<th>Testing Method, EC covered (BV – if applicable)</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Classification of triangle</td>
<td>(1, 3, 3)</td>
<td>ECP: EC1, EC3, EC5, EC7 BV: “1”</td>
<td>isosceles</td>
</tr>
</tbody>
</table>

### Part 1 - Appendix 2: Sample Failure Report

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Functionality Under Test</th>
<th>Description of Failure</th>
<th>Expected Result</th>
<th>Actual Result</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Classification of triangle</td>
<td>Expected &quot;isosceles&quot; but returned &quot;scalene&quot; with input (1, 3, 3)</td>
<td>isosceles</td>
<td>scalene</td>
<td>Severe Problem</td>
</tr>
</tbody>
</table>