ECE 322 Lab Assignment 1

Black Box testing #1

Overview

This laboratory assignment serves as a practical introduction to rudimentary black-box testing techniques. Students employ generic methods such as dirty testing, error guessing, and partition testing to get a feel for test case development and execution.

Introduction to black-box testing

Black-box testing is realized with respect to software specifications where no knowledge of the internal structure or implementation details are known. In this scenario, the application under test is the "black box".

A set of conditions, frequently inputs, is referred to as a test case. Once a test case has been implemented, it is evaluated against the requirements of application under test, and the results are recorded. It may take many different test cases to determine whether the requirements of the system are fully satisfied. Sets of test cases are organized into test suites, where a test suite fulfils some higher level testing requirement.

Although there are an extensive number of black-box testing techniques, in this laboratory experiment we will concentrate on two categories: (i) Failure testing/ error guessing, and (ii) Partition based testing. The following sections will briefly outline each of these categories.

Failure/Dirty Testing

- Test every possible input/action a user can perform on the system to demolish the software
- Think diabolically
- Look at every input
- Be creative; consider the user as someone from another planet
- Examples: divide by zero, wrong input type, missing values

Error Guessing

- Intuition, Innovation, Experience
- Intelligently guess at likely error location
- Use your knowledge as a programmer to explore likely areas of mistakes
- Look for common mistakes, e.g. rounding errors or invalid inputs

Partition Based Testing

- Cover as many possibilities as possible with as few test cases as possible
- Group together inputs for which the application is likely to behave the same
- Test *equivalent* inputs as a group to reduce the number of test cases
- Equivalent inputs are defined in terms of logical partitions of the input space

Equivalence Partitioning is designed to minimize the number of required test cases by forming partitions or equivalence classes of the input space. Test cases are selected in such a way that all equivalence classes are covered. Equivalence classes are sets of inputs for which we expect the system to behave the same, logically, e.g. positive integers, geometric categorizations, numbers of inputs, etc.

Boundary value analysis is a category of testing where we develop test cases that are concerned with the boundaries of equivalence classes. Boundaries are those points at which the behavior of a system is expected to change.

Preparation:
Prepare test cases for the two tasks you will be working on during the lab session. For the calculator, **each test case should include 4 columns:** Test ID (an incrementing number), Description (a brief explanation of the test), Expected Result, and Actual Result. For the **triangle program each test case should include 7 categories:** Test ID, Inputs *a*, *b*, and *c*, Description, Expected Result, Actual Result. In this scenario the values for *a*, *b*, and *c*. During the lab session you should:

- Create test cases for the specified functions of the calculator program
  - For Failure testing
  - For Error guessing
- Identify all equivalence classes for the triangle program's input space
- Formally define the requirements and boundaries of each of these equivalence classes
- Create test cases for the triangle program based on your equivalence classes.

Lab Experiments

Task 1 (35 marks):
For this task you will be testing a calculator application using failure (dirty) testing and error guessing. The application to be tested can be downloaded from the class website.
This application is a simple calculator which implements a number of standard calculator operations. Characters are regarded as invalid input. For this task you will make test cases to cover the provided mathematical functionality of the program. Consider all mathematical operators, invalid inputs, and orders of operation (BEDMAS). Record the results of your testing and identify and errors in the application you find.

Submit your completed test case table **highlighting** the test cases where errors were found.
Comment on which operation, or what mathematical feature is the culprit. *Note that these expectations for test case tables and failure highlighting will be expected for ALL lab assignments in this course.*

Task 2 (35 marks):
For this task you will be testing a command line triangle classification program. The application is available on the class website.
The triangle application reads in three **space separated** integers representing the length of each side of the triangle, and outputs either the type of triangle (scalene, isosceles, or equilateral) or gives an error message. Integers should be entered after the prompt:

```
triangle> a b c
```

For example

```
triangle> 5 5 5
```

Should output

```
Equilateral
```

A more detailed specification for the input and output of the program follows:
Input:

- Takes 3 positive integers
- Only accepts integers
- Three integers on the same line separated by a space character
- Command is executed by pressing enter.

Output:

- Prints out triangle type, one of scalene, equilateral, or isosceles
- Or an error message

Consider equivalence classes for this application, the requirements of each type of triangle, and the error cases. Error cases include problems such as invalid input or the wrong number of arguments. Execute your prepared test cases and submit your results as part of the lab report.
Make sure to **highlight failed test cases** and give an explanation of any faults found in the program. Error guessing and dirty testing do NOT need to be performed on this application.

Lab report:
Must be typed, no handwritten versions will be accepted. Follow the general format as included in the ECE lab guidelines. Your report should include:

- Your write-up
- Definition of your equivalence classes
- The results of your test cases with meaningful descriptions
- Your observations on the effectiveness of each testing technique for identifying errors in the application
- The identification of any errors you may have discovered in the applications during testing, and a brief discussion on what may have caused these failures.

Clearly indicate all failed test cases, and explain the cause of these failures in your write-up.