Introductory Lab Presentation

ECE 322 Lab

Overview

- Introductory Remarks
- Lab Schedule
- Course Expectation
- Report Expectation
- Submission Guidelines
- Lab Topics
- Final Items

ECE 322 Lab Teaching Team

- **Lab Instructor:** Ali Safari Mamaghani

  PhD student in Software Engineering and Intelligent Systems

  Contact: safarima@ualberta.ca

- **Teaching Assistant:** Ahmed Chaari

  PhD student in Software Engineering and Intelligent Systems

  Contact: chaari@ualberta.ca

Lab Schedule

- There are six assignments in this lab component, every other week, starting 17 September 2019

  Lab 1: Sep 17th (70 marks)

  Lab 2: Oct 1th (70 marks)

  Lab 3: Oct 15th (80 marks)

  Lab 4: Oct 29th (100 marks)

  Lab 5: Nov 19th (100 marks)

  Lab 6: Dec 3th (80 marks)

- The six labs in this course are differently weighted toward your final grade .

Course Expectations

- Attendance is mandatory
  - However, you don’t need to be present for the entire duration
  - Expectations for each lab will be made clear
- Preparation
  - Tasks described as *preparation* exercises are “due” at the end of the lab session (4:50 pm)
  - They don’t need to be completed before arriving at the lab
  - This portion of the lab is marked on completion. Failure to complete these sections to the satisfaction of the LI/TA will result in a grade deduction on your report

Lab Reports

- Generally, lab reports should follow ECE report guidelines
- Specific requirements for each lab are given explicitly in the lab manuals

- Each report must include the following sections:
  - An introduction
  - A section for each task including a discussion and test case tables
  - A conclusion
  - Any code or other resources you used in the completion of your lab assignment
  - Any other materials specified in the lab instructions

Lab Reports

- Introduction
  - State the purpose of the lab, what methods are used, what is being tested
  - Convince us that you know what the lab is about, and what it's aim
  - Describe the testing methodologies used in this lab session
- Tasks
  - Provide a description of the task, and the methodology being used
  - Always include **a test case table** with *completed* test cases
  - Highlight interesting results (e.g. failures)
  - Justify your choices (why do you have the tests you do? What does the method you are using accomplish?)
  - Discuss results, failures, and effectiveness
  - Discuss points are often highlighted in the lab manual, read it!

Lab Reports

- Conclusions
  - Note anything interesting you found in your experiments
  - Sum up what you learnt, and how well it went
- Other resources
  - Submit any code written for the lab
  - Submit and provide code which you modified
  - Submit all charts and figures created or used in the lab
  - In general, you should submit *any external resources* used in the completion of the lab
  - Cite your resources!

Test case planning/design

- Example template for a test case table:

Test ID	Description	Expected results	Actual results

Identifier

Set of steps and input conditions

- Test case can lead to success or failure!

Submission Guidelines

- Lab reports are due **one week** after the lab session
  - Tuesdays, by **11.55 pm**
  - Deadline is enforced on Eclass
  - Late assignments are not expected, bring exceptional circumstances.
- Submit your report/code via Eclass
  - Use pdf format for the reports
  - Eclass only allows one file to be uploaded
  - While submitting code projects or other resources you will need to achieve your submission
  - Use .tar .rar or .zip
- Marks will be returned via Eclass before the start of the following lab.

Lab Topics

Lab 1

- 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.

- Three types of simple black-box testing are covered in lab 1:
  - Failure/Dirty Testing
  - Error Guessing
  - Partition-based Testing (and Boundary Value Analysis)

Lab 1

- 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

Lab 1

- 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

Sorting subroutine

Test cases
•Input list is empty
•Input list contains one entry
•All entries in the input have the same value
•The input is already sorted

Lab 1

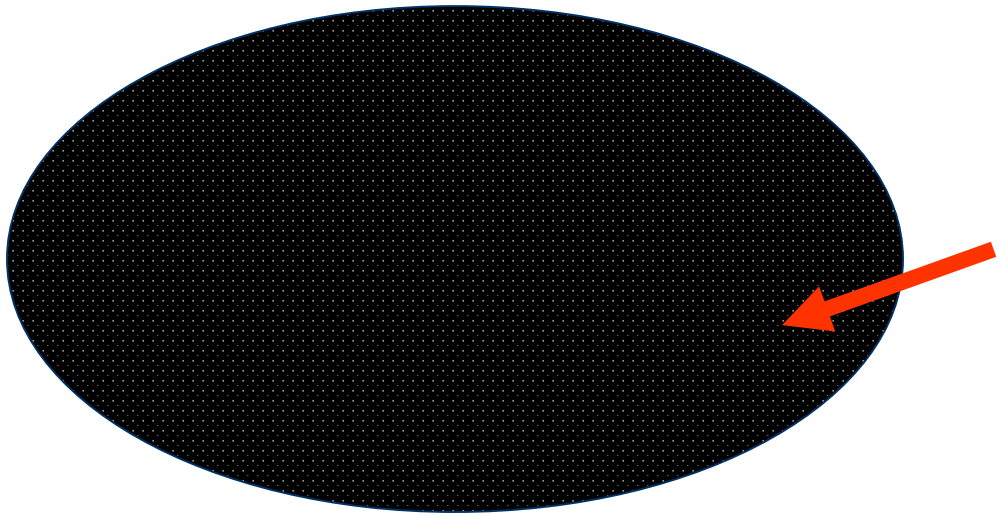
- Partition-based Testing
  - Cover as many possibilities as possible with as few test cases as possible
  - Group together inputs for which the application is likely to behave the same
  - Test equivalent inputs as a group to reduce the number of test cases
  - Equivalent inputs are defined in terms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

- Equivalence Partitioning
  - 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.

Partition coverage- example 1

Equivalence Partitioning: Example

$$ax^2 + bx + c = 0$$



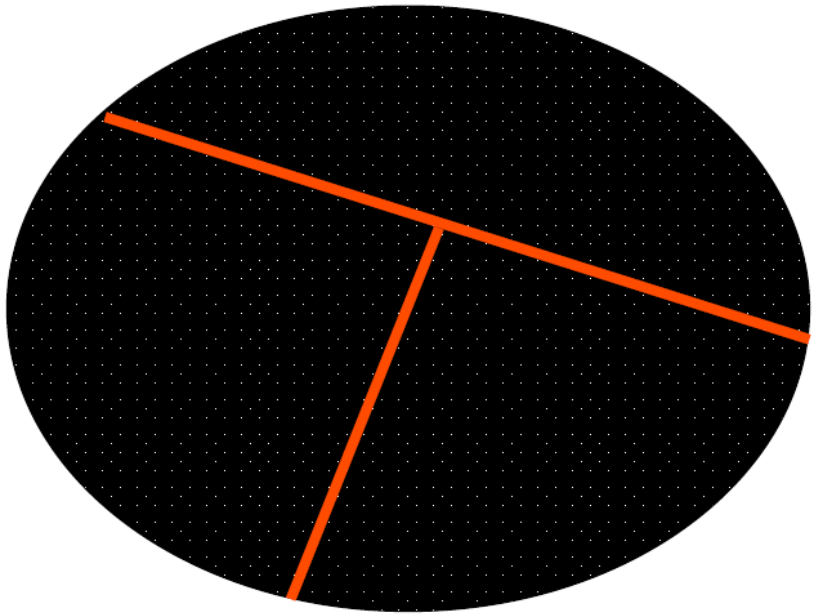
a, b, c - 32 bit numbers

Space of 32 bit numbers {a, b, c}

a, b, c - 32 bit numbers→ $2^{32} * 2^{32} * 2^{32} = 2^{96}$ test cases

Lab 1

- Equivalence Partitioning: Example



$ax^2 + bx + c = 0$

$d = b^2 - 4ac$

Partition – 3 test cases
d = 0
d >0
d <0

Lab 1

- Equivalence Classes
  - Two types of classes: valid equivalence classes, and invalid equivalence classes
- Test case design
  1. Identification of equivalence classes
  2. Defining the test cases
- Defining the test cases
  1. Assign a unique number to each equivalence class
  2. Until all **valid equivalence classes** have been covered by test cases, write a new test case covering as many of the uncovered valid equivalence classes as possible
  3. Until tests have covered all **invalid equivalence classes**, write a test case that covers one and only one, of the uncovered invalid equivalence classes

Lab 1

Equivalence Classes: Example

- Input: range of values, 1…999 (item count)

  Valid equivalence class 1…999
  Invalid equivalence class ″less than 1″, ″greater than 999″

- One through six owners (insurance policy)

  Valid equivalence class 1..6
  Invalid equivalence class 0 >6

- First character of the identifier must be a letter

  Valid equivalence class: letter
  Invalid equivalence class: not a letter

Lab 1

Equivalence Classes: Example

DIM statement **DIM name[, , , …]**

(a) Array name: up to six letters or digits, the first must be letter

(b) Number of dimensions 1, 2,…, 7

Input condition	Valid equivalence classes	Invalid equivalence classes	
Length of array name	1-6 (1)	0 (2)	>6 (3)
Starts with letter	yes (4)	no (5)	
Number of dimensions	1-7 (6)	0 (7)	>7 (8)

Lab 1

Equivalence Classes: Example

Input condition	Valid equivalence classes	Invalid equivalence classes
Length of array name	1-6 (1)	0 (2) >6 (3)
Starts with letter	yes (4)	no (5)
Number of dimensions	1-7 (6)	0 (7) >7 (8)

Test cases to cover all equivalence classes

DIM A[2] covers (1) (4) (6) DIM 1B[4] covers (5)

DIM D[] covers (7) DIM [10, 30, 11, 12, 13, 14, 45] covers (2)

DIM DATA[100, 110, 20, 50, 90, 200, 70, 100] covers (8)

DIM DATASENSOR1[100, 100] covers (3)

Lab 1

- 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 behaviour of a system is expected to change.

Lab 2

- Second Black-box testing lab, more formally structured methods
- Two formal methods will be examined:
  - Extreme Point Combination (EPC)
  - Weak n×1 strategy
- Encourage the use of test case automation

Lab 3

- Third (and final) Black-box testing lab
  - Cause-effect graphs
  - Decision tables
  - Logical System rules
  - Test case reduction

Lab 4

- White-box testing #1
- Junit testing
  - Control Flow Graphs
  - Code coverage
  - Junit testing
- Pairwise Testing and Orthogonal Arrays
  - Comparison between procedural tools and known arrays
  - PICT pairwise test case generation tool

Lab 5

- Advanced White-box testing
- Integration testing
  - Top Down
  - Bottom Up
  - Big Bang

Lab 6

- Final White-box testing
- Regression testing
  - Code Maintenance with Unit Testing
  - Catching induced errors from added features and fixing other bugs
- Mutation testing
  - Testing your tests
  - PiTest Library

Contact and Office Hours

- No set weekly office hours
  - ECE Graduate offices are not accessible to students
  - Meeting can be arranged via email if assistance is needed
- Email contact:
  - safarima@ualberta.ca (Ali)
  - chaari@ualberta.ca (Ahmed)
- We will try to answer all emails within 24h
  - No guarantees
  - Be mindful of deadlines and exam dates

Final Items

- All submissions will be online through Eclass
  - There is no drop box for the lab
- First lab is on September 17th
  - Lab starts at 2 pm
  - 5th floor of ETLS in room E5- 005

Questions?