Software Testing

Types of Testing Techniques & Test Case Design

Lesson Objectives

- **To understand the following topics:**
  - Types of testing techniques
  - Static testing - Review, Code inspection
  - Dynamic testing - White Box, Black Box.
  - Test Case Construction
  - Test Data Preparation

Types

- **Static Testing - Testing a software without execution on a computer. Involves just examination/review and evaluation.**
  - **Reviews (of Code and Imp. Documents like Req., Design)**
    - **Self Review**
    - **Peer Review**
    - **Senior Review**
    - **Group Review**
  - **Code Inspection**

- **Dynamic Testing - Testing software through executing it.**

Introduction

- **Static Testing is a process of reviewing the work product and reviewing is done using a checklist.**
- **Static Testing helps weed out many errors/bugs at an early stage**
- **Static Testing lays strict emphasis on conforming to specifications.**
- **Static Testing can discover dead codes, infinite loops, uninitialized and unused variables, standard violations and is effective in finding 30-70% of errors**

Comparison

White Box Test Techniques

- Code Coverage
    - Statement Coverage
    - Decision Coverage
    - Condition Coverage
- Memory Leakage

Black Box Test Techniques

- Equivalence Partitioning
- Boundary Value Analysis
- Error Guessing

2.3: Dynamic Testing

White Box Test Techniques

- White box is logic driven testing and permits Test Engineer to examine the internal structure of the program
- Examine paths in the implementation
- Make sure that each statement, decision branch, or path is tested with at least one test case
- Desirable to use tools to analyze and track Coverage
- White box testing is also known as structural, glass-box and clear-box.

Statement Coverage

- **Test cases must be such that all statements in the program is traversed at least once**

- **Consider the following snippet of code**

  void procedure(int a, int b, int x)

```
{       If (a>1) && (b==0)
                  { x=x/a; }
        If (a==2 || x>1)
                  {  x=x+1; }
}
```

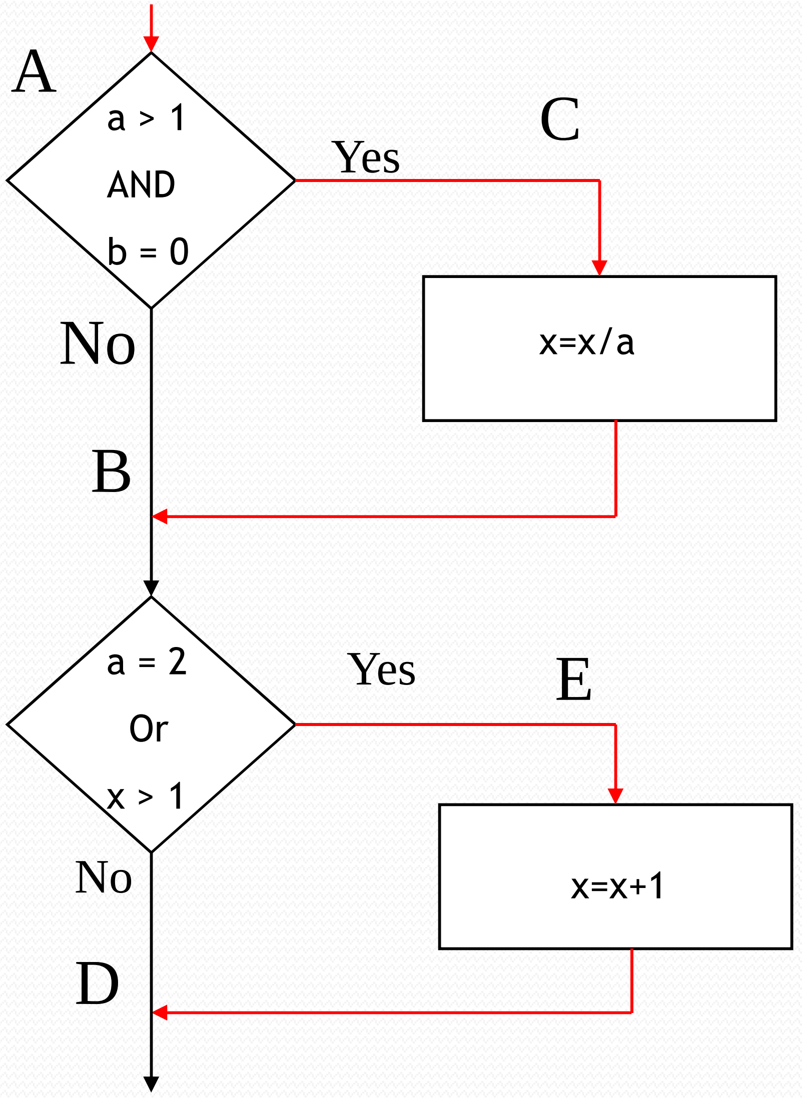
Statement Coverage

Test Case: a=2,b=0, x=3.

Every statement will be executed once.

But only path ACE will be covered and path

ABD,ACD,ABE will not be covered.



2.3: Dynamic Testing

Statement Coverage

- In the above code one test case is sufficient to execute each of the two if statements at least once:

  Test Case : a=2 , b=0 , x=3

  (Decision1 is True, Decision2 is True)

  However this test case does not help in detecting many of the many of the bugs which may go unnoticed as the false outcomes of the conditions a>1 & b=0 , a=2 or x>1 are not tested.

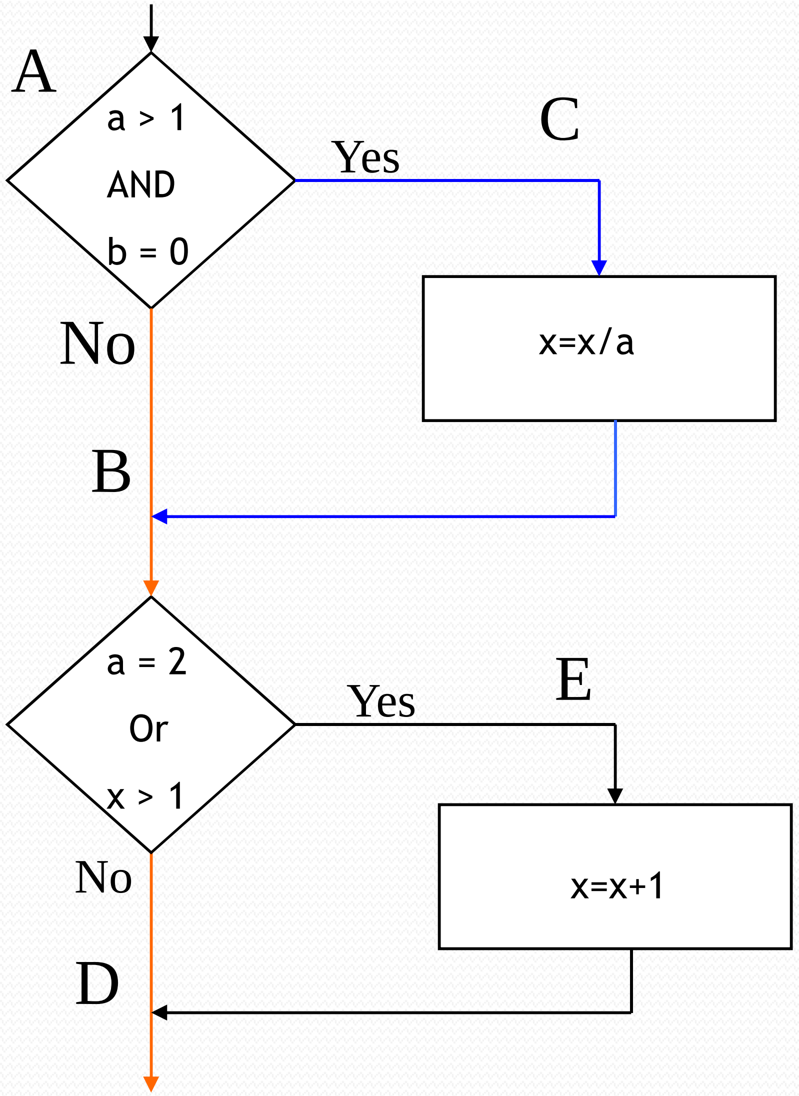
Decision Coverage

Test Case 1: a=2, b=0, x>1

(Decision1 is True, Decision2 is True) (Path ACE)

Test Case 2: a<=1 , b!=0, x<=1

(Decision1 is False, Decision2 is False) (Path ABD)

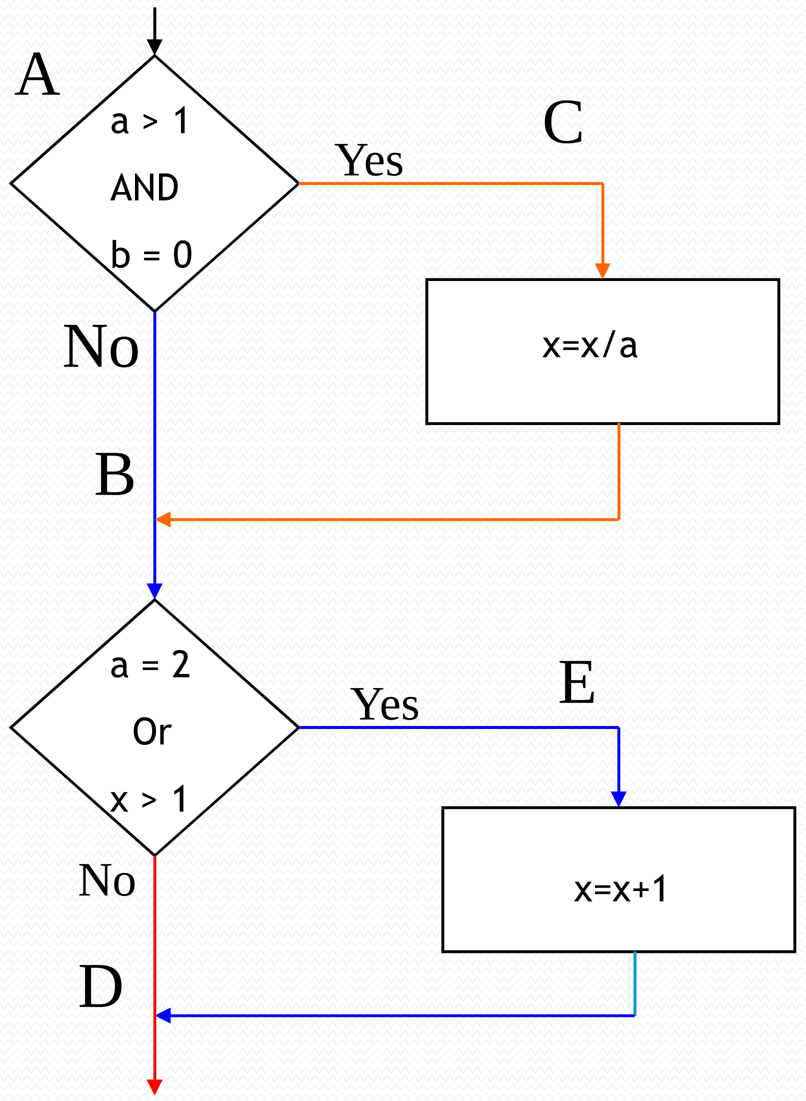


Condition Coverage

- **Test cases are written such that each condition in a decision takes on all possible outcomes at least once.**

Test Case1 : a=2, b=0, x=3
(Condition1 is True,Condn2 is True)
(Path ACE)

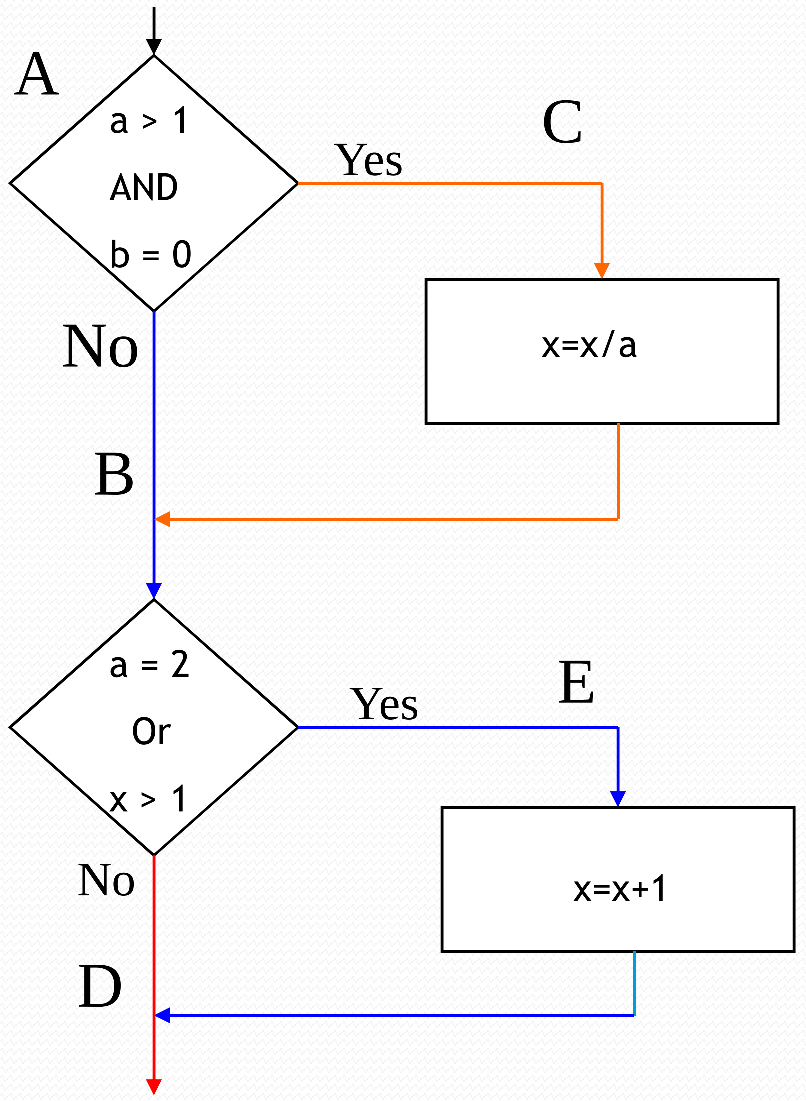
Test Case2: a=3, b=0, x=0
(Condn1 is True,Condn2 is False)
(Path ACD)



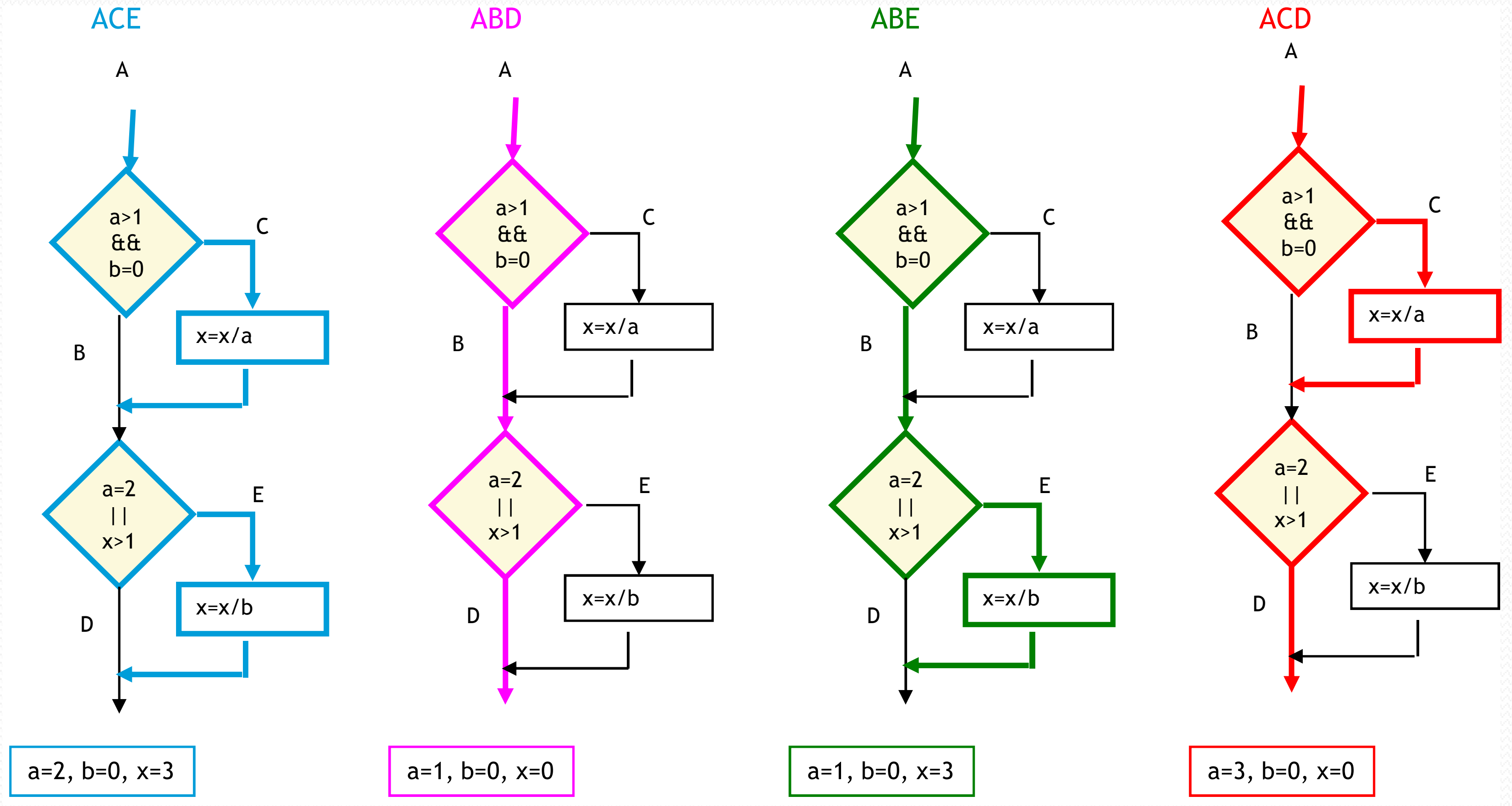
Condition Coverage

Test Case3 : a=1, b=0, x=3
(Condition1 is False,Condition2 is True)
(Path ABE)

Test Case4: a=1, b=1, x=1
(Condition1 is False,Condition2 is False)
(Path ABD)



Condition Coverage



Memory Leak

- **Memory leak is present whenever a program loses track of memory.**
- **Memory leaks are most common types of defect and difficult to detect**
- **Performance degradation or a deadlock condition occurs**
- **Memory leak detection tools help to identify**
  - memory allocated but not deallocated
  - uninitialized memory locations

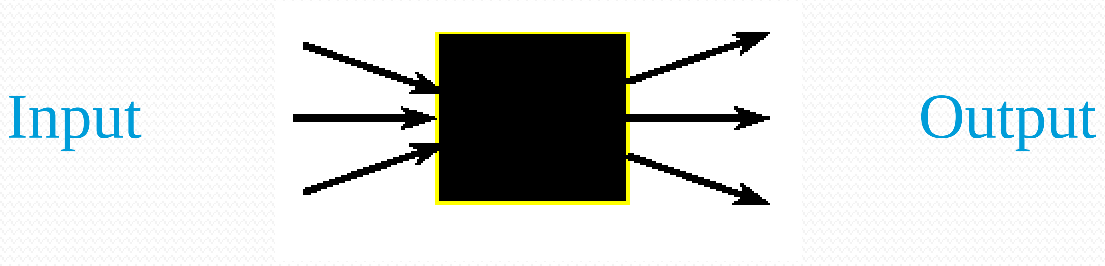
Memory Leak

- **Find the error in the following snippet of code**

```
void read_file(char*);
void test(bool flag)
{
        char* buf = new char[100];
         if (flag) {
                    read_file(buf);
                    delete [] buf;
                    }
 }
```

Black Box Test Techniques

- **Black box is data-driven, or input/output-driven testing**
- **The Test Engineer is completely unconcerned about the internal behavior and structure of program**
- **Black box testing is also known as behavioral, functional, opaque-box and closed-box.**



Black Box Test Techniques

Tests are designed to answer the following questions:

- **How is functional validity tested ?**
- **What classes of input will make good test cases?**
- **Is the system particularly sensitive to certain input values?**
- **What effect will specific combinations of data have on system operations?**

Black Box Test Techniques

- **Equivalence Partitioning**
- **Boundary Value Analysis**
- **Error Guessing**

Equivalence Partitioning

- **This method divides the input domain of a program into categories of data for deriving test cases.**

- **Identify equivalence classes - the input ranges which are treated the same by the software**
  - Valid classes: legal input ranges
  - Invalid classes: illegal or out of range input values

- **The aim is to group and minimize the number of test cases required to cover these input conditions**

Equivalence Partitioning

Assumption:

- **If one value in a group works, all will work**

  One from each partition is better than all from one

- **Thus it consists of two steps:**
  - Identify the Equivalence class
  - Write test cases for each class

Equivalence Partitioning

Examples of types of equivalence classes

- **1. If an input condition specifies a continuous range of values, there is one valid class and two invalid classes**

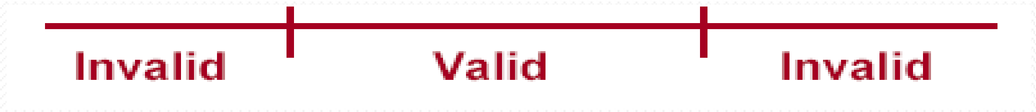
  Example: The input variable is a mortgage applicant's income. The valid range is $1000/mo. to $75,000/mo.

  - Valid class: {1000 > = income < = 75,000}

  - Invalid classes: {income < 1000}, {income > 75,000}

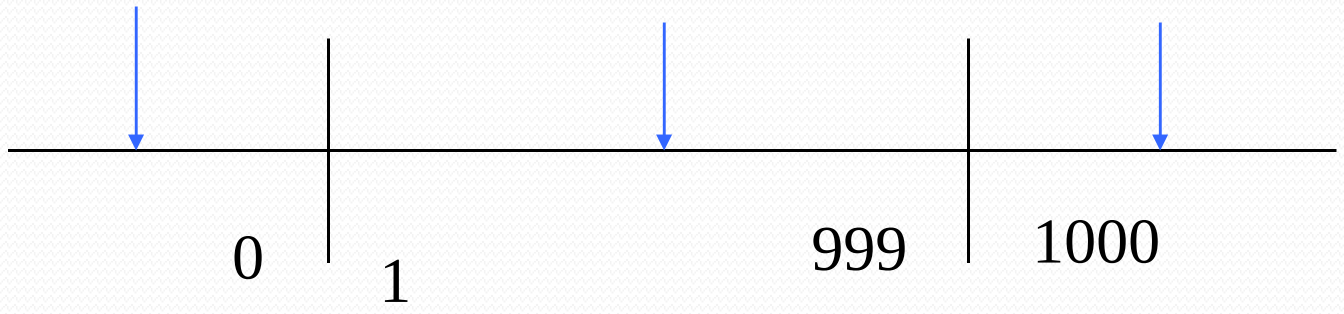
Equivalence Partitioning

2. If an input condition specifies that a variable, say count, can take range of values(1 - 999),



Identify - one valid equivalence class (1 <= count <= 999)

- two invalid equivalence classes (count < 1) & (count >999)



Equivalence Partitioning

3. If a “must be” condition is required, there is one valid equivalence class and one invalid class

Example: The mortgage applicant must be a person.

- Valid class: {person}
- Invalid classes:{corporation, ...anything else...}

Equivalence Partitioning

Example

If we have to test function int Max(int a , int b) the Equivalence Classes for the arguments of the functions will be

Arguments	Valid Values	Invalid Values
A	-32768 <= Value <= 32767	< - 32768 , >32767
B	-32768 <= Value <= 32767	< - 32768 , >32767

Boundary Value Analysis

- “Bugs lurk in corners and congregate at boundaries …..”

  Boris Beizer

- Boundary Conditions are those situations directly on, above, and beneath the edges of input equivalence classes and output equivalence classes.

- Boundary value analysis is a test case design technique that complements Equivalence partitioning

- Test cases at the boundary of each input Includes the values at the boundary, just below the boundary and just above the boundary

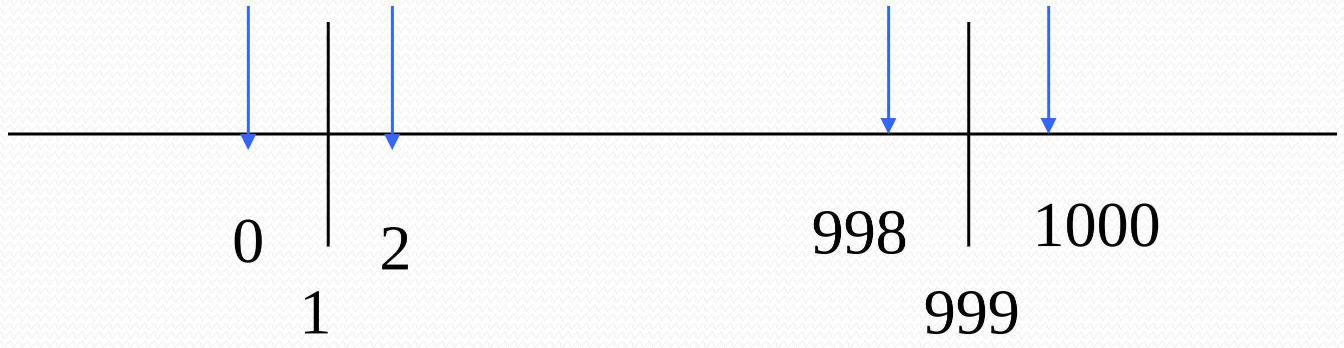
Boundary Value Analysis

From previous example, we have the valid equivalence class as (1 <= count < = 999).

Now, according to boundary value analysis,

we need to write test cases for

count=0, count=1,count=2,count=998,count=999 and count=1000 respectively



Boundary Value Analysis

Guidelines

- If an input condition specifies a range of values A and B, test cases should be designed with values A and B, just above and just below A and B respectively
- Similarly with a number of values

Error Guessing

- **Based on experience and intuition one may add more test cases to those derived by following other methodologies.**
- **It is an ad hoc approach**
- **The basis behind this approach is in general people have the knack of “smelling out” errors**

Error Guessing

- **Example : Suppose we have to test the login screen of an application. An experienced test engineer may immediately see if the password typed in the password field can be copied to a text field which may cause a breach in the security of the application.**

- **Error guessing testing for sorting subroutine situations**
  - The input list empty
  - The input list contains only one entry
  - All entries in the list have the same value
  - Already sorted input list

Introduction

- **Test cases construction and test data preparation are the first stages of testing stage.**
- **Test cases are prepared based on test ideas**
- **“A test idea is a brief statement of something that should be tested. “**
  - For example, if you're testing a square root function, one idea for a test would be ‘test a number less than zero’.
- **“ The idea of preparing a test case is to check if the code handles an error case.”**

Test Case

- **A set of inputs, execution preconditions, and expected outcomes developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.**

- **Test cases may be designed based on**
  - Values – Valid/Invalid/Boundary/Negative
  - Test conditions

- **Test case will be complex if there is more than one expected result.**

Test Case Terminologies

- **Test Scenario**
  - It is an end-to-end flow of a combination of test conditions & test cases integrated in a logical sequence, covering a business processes

- **Test Condition / Pre Condition**
  - Environmental and state which must be fulfilled before the component/unit can be executed with a particular input value.
  - It is a set of rules under which a tester will determine if a requirement is partially or fully satisfied
  - One test condition will have multiple test cases

Test Case Terminologies

- **Test Data/Input**
  - Inputs & its combinations / variables used
- **Test Procedure**
  - A detailed description of steps to execute the test
- **Test Result/Output**
  - Pass / Fail – If the program works as given in the specification, it is said to Pass otherwise Fail.
  - Failed test cases may lead to code rework

Test Case Lifecycle

- **Write an effective and efficient set of test cases.**
- **The set of tests should find common bugs.**
- **The set of tests finds those bugs with a reasonable amount of effort.**
- **Test case incorporated tester action, data, and expected result. The action is usually to input some data, but it might be other actions as well.**
- **A template is usually followed for writing test cases.**

Test Case Design Technique

- **Test cases are designed based on the following techniques**
  - Specification-based - Black Box testing techniques
    - Boundary value analysis, Equivalence partitioning
  - Structure – based – White Box testing techniques
    - Code coverage, Decision coverage, Condition coverage

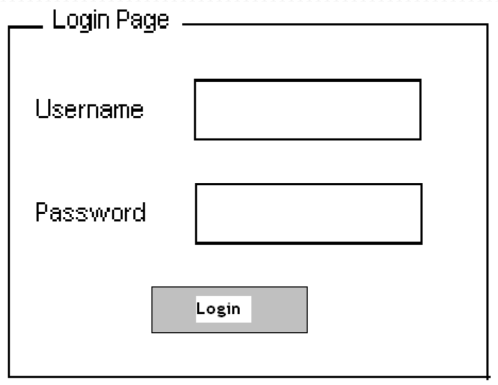
Test Case Template

Assumptions:

	Test Case ID	PreConditions (If any)	Test Condition / Scenario	Test Steps	Input /Test Data	Expected Result	Actual Result

Test Case Example

- **Writing a testing case to test the following**
  - A login form takes user name and password data as input in two text boxes
  - Checks for valid user name and password
  - Enters home page if user name and password is valid
  - Else it displays error message



Test Case Example

- **Test Preconditions:**
  - A list of valid user name and password is available
    - Example:
      - Username: Jim
      - Password: pass123

      - Username: Harry
      - Password:mypass123

Test Case Example

- **Test Post Conditions:**
  - The user is either logged in and home page is displayed
  - Error message is displayed if either the username or password is incorrect.
- **Test Objective:**
  - To check whether the entered username and password are valid or invalid.
    - Home page is displayed for valid inputs.
    - Error message is displayed for invalid inputs.

Test Case Design Technique

Assumptions: Valid UserName and password are available in database

	Test Case ID	**Preconditions (If any)**	**Test Condition / Scenario**	**Test Steps**	**Input /Test Data**	**Expected Result**	**Actual Result**
	TC_01	The application is invoked and login page is displayed	To validate the login page with null password	Enter User Name	User Name= Mark		
				Enter password	Password=		
				Press LOGIN Button		Should Display Warning Message Box "Please Enter Valid User name and Password"	
	TC_02	The application is invoked and login page is displayed	To validate the login page with null username	Enter User Name	User name =		
				Enter password	Password=Accenture1234		
				Press LOGIN Button		Should Display Warning Message Box "Please Enter Valid User name and Password"	
	TC_03	The application is invoked and login page is displayed	To validate the login page with valid username and invalid password	Enter User Name	User Name= COES		
				Enter password	Password= XYZ		
				Press LOGIN Button		Should Display Warning Message Box "Please Enter Valid User name and Password"	

Test Case Design Technique

	TC_04	The application is invoked and login page is displayed	To validate the login page with invalid username and valid password	Enter User Name	User Name= XYZ		
				Enter password	Password= COES		
				Press LOGIN Button		Should Display Warning Message Box "Please Enter Valid User name and Password"	
	TC_05	The application is invoked and login page is displayed	To validate the login page with invalid username and invalid password	Enter User Name	User Name= XYZ		
				Enter password	Password= XYZ		
				Press LOGIN Button		Should Display Warning Message Box "Please Enter Valid User name and Password"	
	TC_06	The application is invoked and login page is displayed	To validate the login page with null username and null password	Enter User Name	User Name=		
				Enter password	Password=		
				Press LOGIN Button		Should Display Warning Message Box "Please Enter Valid User name and Password"	
	TC_07	The application is invoked and login page is displayed	To validate the login page with valid username and password	Enter User Name	User Name= JIM		
				Enter password	Password= pass123		
				Press LOGIN Button		Should navigate to the Home page	

What is test data?

- **Test Data**
  - An application is built for a business purpose. We input data and there is a corresponding output. While an application is being tested we need to use dummy data to simulate the business workflows. This is called test data.
  - A test scenario will always have an associated test data.Tester may provide test data at the time of executing the test cases or application may pick the required input data from the predefined data locations.
  - The test data may be any kind of input to application, any kind of file that is loaded by the application or entries read from the database tables. It may be in any format like xml test data, stand alone variables, SQL test data etc

📖 ***If you are testing with bad or unstable data, how can you be sure your test results are accurate!!!***