Terna Engineering College

Computer Engineering Department

Program: Sem VI

Course: Software Engineering Lab

LAB Manual

PART A

(PART A: TO BE REFERRED BY STUDENTS)

Experiment No.12

A.1 Aim:

Perform white-box (Basic Path Testing) or black-box testing on anyone function developed (code) of the selected mini project and develop the test cases for same.

A.2 Prerequisite:

Knowledge about software testing and types of testing.

A.3 Outcome:

After successful completion of this experiment students will be able to differentiate between white and black box testing and able to perform white box testing.

A.4 Theory:

- **Software Testing**

Software testing is a method to check whether the actual software product matches expected requirements and to ensure that the software product is **Defect free**. It involves the execution of software/system components **using manual or automated tools** to evaluate one or more properties of interest. **The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.**

- **Testing in Software Engineering**

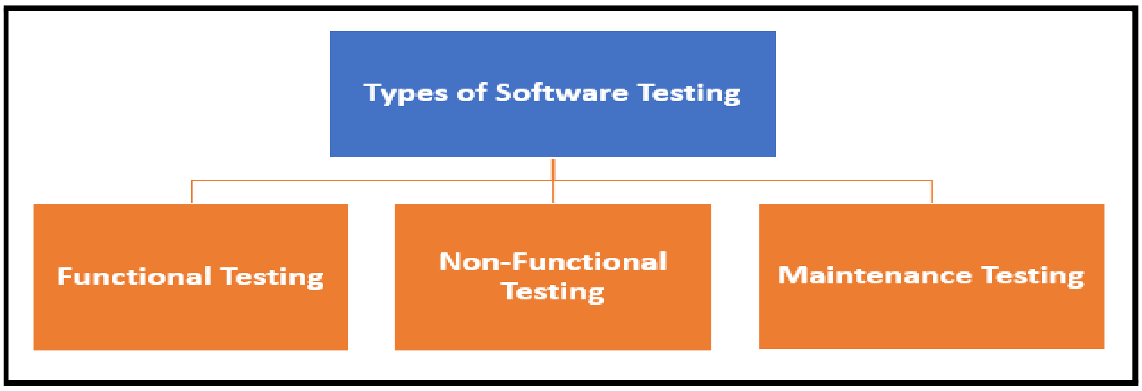
As per ANSI/IEEE 1059, **Testing in Software Engineering is a process of evaluating a software product to find whether the current software product meets the required conditions or not.** The testing process involves evaluating the features of the software product for requirements in terms of any **missing requirements, bugs or errors, security, reliability and performance.**

- Software testing is defined as an activity **to check whether the actual results match the expected results and to ensure that the software system is Defect free.**
- Testing is important **because software bugs could be expensive or even dangerous.**
- The important are reasons for using software testing are **cost-effectiveness, security, product quality, and customer satisfaction.**
- Typically Testing is classified into **three categories** **functional testing, non-functional testing or performance testing, and maintenance.**
- The important strategies in software engineering are **unit testing, integration testing, system testing, validation testing (User acceptance testing).**

➢ **Types of Software Testing**

Typically Testing is classified into three categories.

- **Functional Testing**
- **Non-Functional Testing or Performance Testing**
- **Maintenance (Regression and Maintenance)**



➢ Testing Strategies (levels) in Software Engineering

Here are important testing strategies in software engineering:

Unit Testing: This software testing approach is followed by the programmer to test the unit of the program. It helps developers to know whether the individual unit of the code is working properly or not.

Integration testing: It focuses on the construction and design of the software. You need to see that the integrated units are working without errors or not.

System testing: In this method, your software is compiled as a whole and then tested as a whole. This testing strategy checks the functionality, security, portability, amongst others.

User Acceptance Testing (Validation Testing):

➢ Functional Testing

1. What is Functional Testing?

FUNCTIONAL TESTING is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Functional testing mainly involves <u>black box testing</u> and **it is not concerned about the source code of the application**. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application under Test. **The testing can be done either manually or using automation**.

What do you test in Functional Testing?

The prime objective of Functional testing is **checking the functionalities of the software system.** It mainly concentrates on -

- **Mainline functions**: Testing the main functions of an application
- **Basic Usability**: It involves basic usability testing of the system. It checks whether a user can freely navigate through the screens without any difficulties.

- **Accessibility**: Checks the accessibility of the system for the user
- **Error Conditions**: Usage of testing techniques to check for error conditions. It checks whether suitable error messages are displayed.

How to do Functional Testing

- Understand the Functional Requirements
- Identify test input or test data based on requirements
- Compute the expected outcomes with selected test input values
- Execute test cases
- Compare actual and computed expected results

➢ Types of Functional Testing

- **White-Box Testing**
    1. Basic Path Testing
    2. Control structure Testing
        A. **Branch Testing**
        B. **Condition Testing**
        C. **Data Flow Testing**
        D. **Loop Testing**
- **Black box testing**
    1. **Equivalence Class Testing (Equivalence Partitioning)**
    2. **Boundary Value Testing**
- **UAT (User Acceptance Testing)**

White-box testing:

- Knowing the internal workings of a product, **test that all internal operations are performed according to specifications** and all internal components have been exercised.
- **Types of White Box Testing:**
    1. **Basic Path Testing**
    2. **Control Structure Testing**

1. Basic Path Testing:

Introduction:

Path testing is a structural testing method **that involves using the source code of a program to find every possible executable path**. It helps to determine all faults lying within a piece of code. This method is designed to execute all or selected path through a computer program.

Any software program includes multiple entries and exit points. Testing each of these points is challenging as well as time-consuming. To reduce the redundant tests and to achieve maximum test coverage, basis path testing is used.

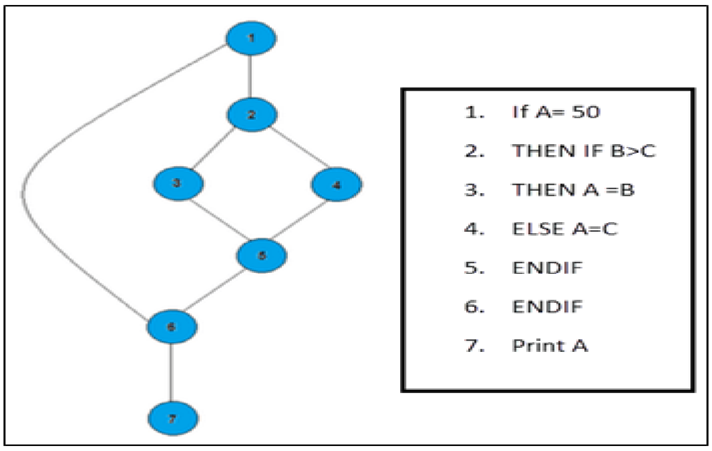
It is a White-box testing technique proposed by Tom McCabe. Enables the test case designer to **derive a logical complexity of a procedural design(system).** And then use this measure **to define the basis set of execution paths the developer uses**. Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least one time during testing.

Basis Path Testing in software engineering is a White Box Testing method in which test cases are defined based on flows or logical paths that can be taken through the program.

The objective of basis path testing:

Is to define the number of independent paths, so the number of test cases needed can be defined explicitly to maximize test coverage.

Here we will take a simple example, to get a better idea of what is the basis of path testing include



```
1.  If A= 50
2.  THEN IF B>C
3.  THEN A =B
4.  ELSE A=C
5.  ENDIF
6.  ENDIF
7.  Print A
```

In the above example, we can see few conditional statements is executed depending on what condition it suffice. Here there are 3 paths or condition that needs to be tested to get the output,

- **Path 1**: 1,2,3,5,6, 7
- **Path 2**: 1,2,4,5,6, 7
- **Path 3**: 1, 6, 7

Steps for Basis Path testing

The basic steps involved in basis path testing include;

- **Draw a control graph (flow graph)** (to determine different program paths)
- Calculate Cyclomatic complexity (metrics to **determine the number of independent paths)**
- **Find a basic set of paths**
- **Generate test cases to exercise each path**

Flow Graph Notation:

- ✔ A circle in a graph represents a node, which stands for a sequence of one or more procedural statements.
- ✔ A node containing a simple conditional expression is referred to as a predicate node.
- ✔ Each compound condition in a conditional expression containing one or more Boolean operators (e.g., and, or) is represented by a separate predicate node.
- ✔ A predicate node has two edges leading out from it (True and False).
- ✔ An edge, or a link, is an arrow representing the flow of control in a specific direction.
- ✔ An edge must start and terminate at a node.
- ✔ An edge does not intersect or cross over another edge.
- ✔ Areas bounded by a set of edges and nodes are called regions.
- ✔ When counting regions, including the area outside the graph as a region too.

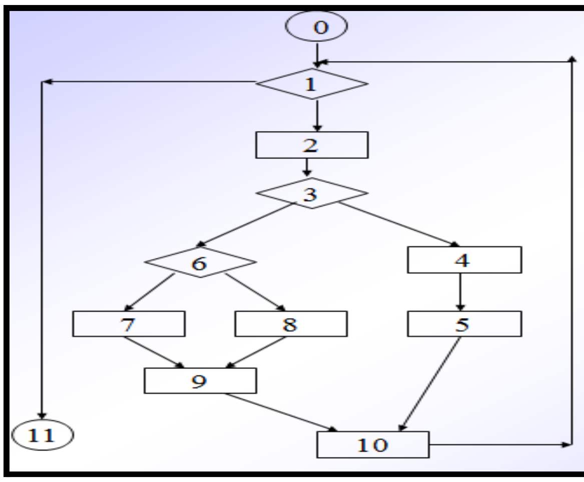


Figure 7.1: Procedural Design using Flowchart

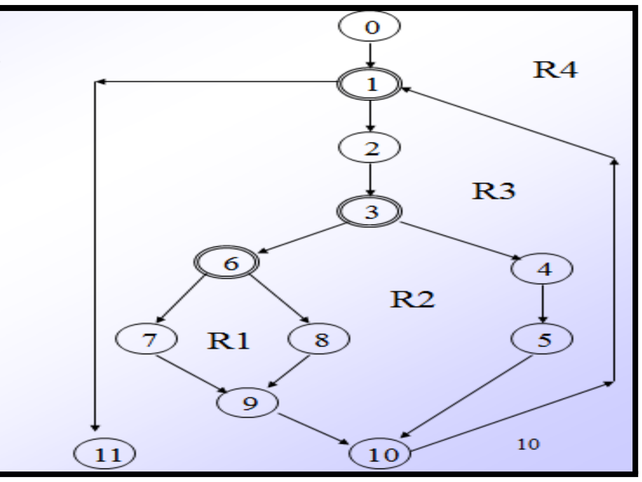


Figure 7.2: Flow Graph Notation of given Flowchart

Independent Program Path:

- ✔ Defined as a path through the program from the start node until the end node that introduces at least one new set of processing statements or a new condition (i.e., new nodes)
- ✔ Must move along at least one edge that has not been traversed before by a previous path
- ✔ Basis set for flow graph on previous slide
  - **Path 1: 0-1-11**
  - **Path 2: 0-1-2-3-4-5-10-1-11**
  - **Path 3: 0-1-2-3-6-8-9-10-1-11**
  - **Path 4: 0-1-2-3-6-7-9-10-1-11**
- ✔ The number of paths in the basis set is determined by the **cyclomatic complexity**
  - o **cyclomatic complexity Can be computed in three ways**
    - ✔ **The number of regions**
    - ✔ V(G) = E – N + 2, where E is the number of edges and N is the number of nodes in graph G
    - ✔ V(G) = P + 1, where P is the number of predicate nodes in the flow graph G
    - ✔ **Number of regions**: Bounded + unbounded rejoins of the flow graph

Results in the following equations for the example flow graph

- Number of regions = 4 (R1,R2,R3 and R4)
- V(G) = 14 edges – 12 nodes + 2 = 4
- V(G) = 3 predicate nodes + 1 = 4
- **Therefore, No. of Independent path = 4 (For Given Flow graph)**
- **Therefore, Prepare test cases that will force execution of all 4 paths**.

Advantages of Basic Path Testing

- It helps to reduce the redundant tests
- It focuses attention on program logic
- It helps facilitates analytical versus arbitrary case design
- Test cases that exercise basis set will execute every statement in a program at least once.

Example 2: Consider the given program that checks if a number is prime or not. For the following program:

```
int main()
{
 1 int n, index;
2  cout << "Enter a number: " <> n;
3    index = 2;
4    while (index <= n - 1)
5    {
6       if (n % index == 0)
7       {
8          cout << "It is not a prime number" << endl;
9          break;
10      }
11      index++;
12   }
13   if (index == n)
14      cout << "It is a prime number" << endl;
15  } // end main
```

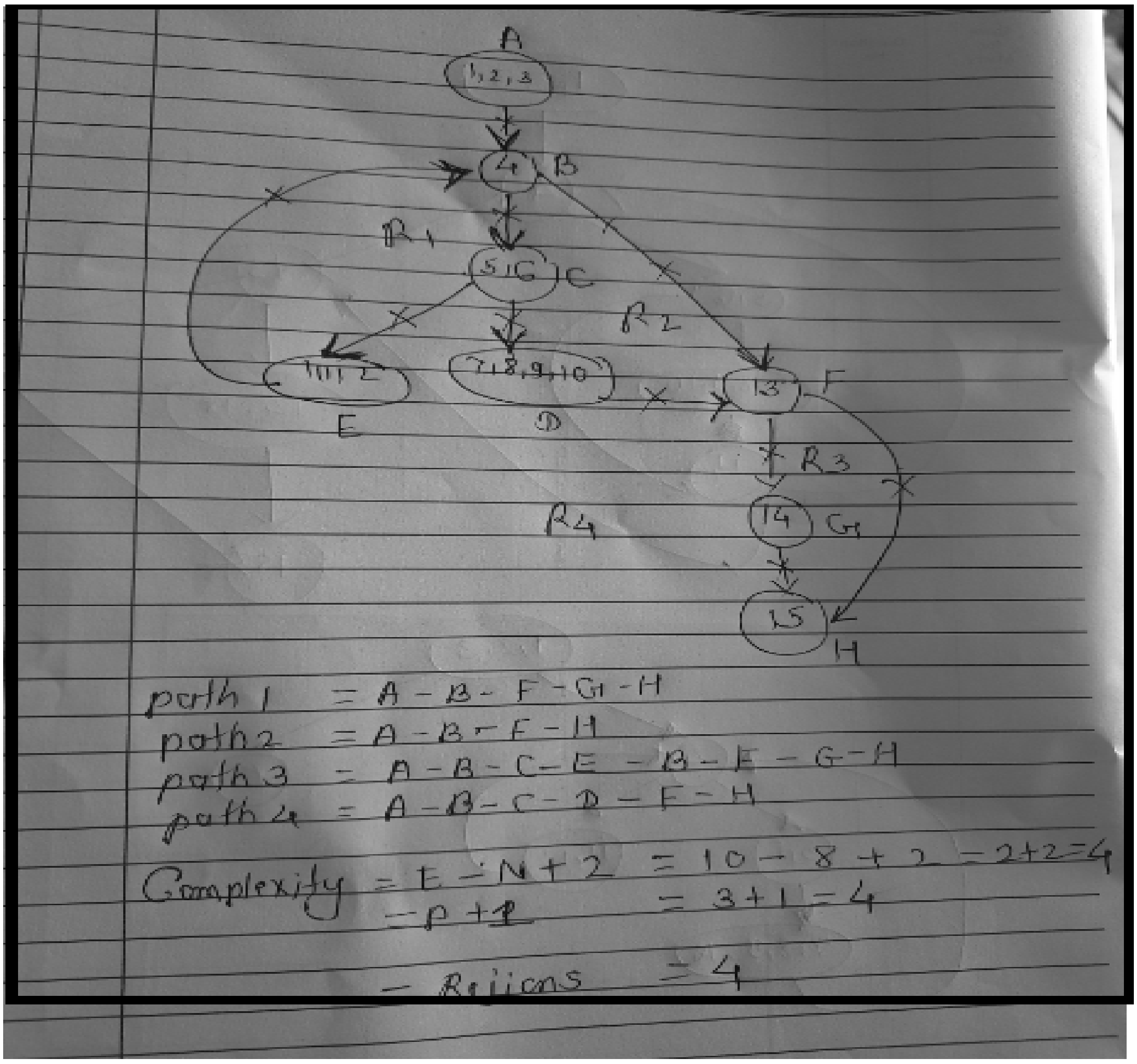
---

1 Draw the Control Flow Graph
2 Calculate the Cyclomatic complexity using all the methods

3 List all the Independent Paths

4 Design test cases from independent paths

1)Flow graph Notation



path 1 = A - B - F - G - H
path 2 = A - B - F - H
path 3 = A - B - C - E - B - F - G - H
path 4 = A - B - C - D - F - H

Complexity = E - N + 2 = 10 - 8 + 2 = 2+2=4
= p + 1 = 3 + 1 = 4
= Regions = 4

1) Cyclomatic complexity

Method-1:
V(G) = e - n + 2
In the above control flow graph,
where, e = 10, n = 8 and p = 1
Therefore,
Cy-clomatic Complexity V(G)
= 10 - 8 + 2
= 4

Method-2:
V(G) = P+1
In the above control flow graph,
where, P = 3 (Node B, C and F)
Therefore,
Cyclomatic Complexity V(G)
= 3 + 1
= 4

Method-3:
V(G) = Number of Regions
In the above control flow graph, there are 4

2) Independent Paths:

Path 1 : A - B - F - G - H

Path 2 : A - B - F - H

Path 3 : A - B - C - E - B - F - G - H

Path 4 : A - B - C - D - F - H

Note:

- ✔ Independent paths are not necessarily unique.
- ✔ **Each of these paths has introduced at least one new edge which has not been traversed before.**

3) **Test cases:**

To derive test cases, we have to use the independent paths obtained previously. To design a test case, provide input to the program such that each independent path is executed.
For the given program, the following test cases will be obtained:

Test case ID	Input Number	Output	Independent path covered
1	1	No output	A-B-F-H
2	2	It is a prime number	A-B-F-G-H
3	3	It is a prime number	A-B-C-E-B-F-G-H
4	4	It is not a prime number	A-B-C-D-F-H

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per the following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Blackboard access available)

Roll No. 50	**Name:** AMEY THAKUR
Class: Comps TE B	**Batch:** B3
Date of Experiment: 28/04/2021	**Date of Submission:** 28/04/2021
Grade:	

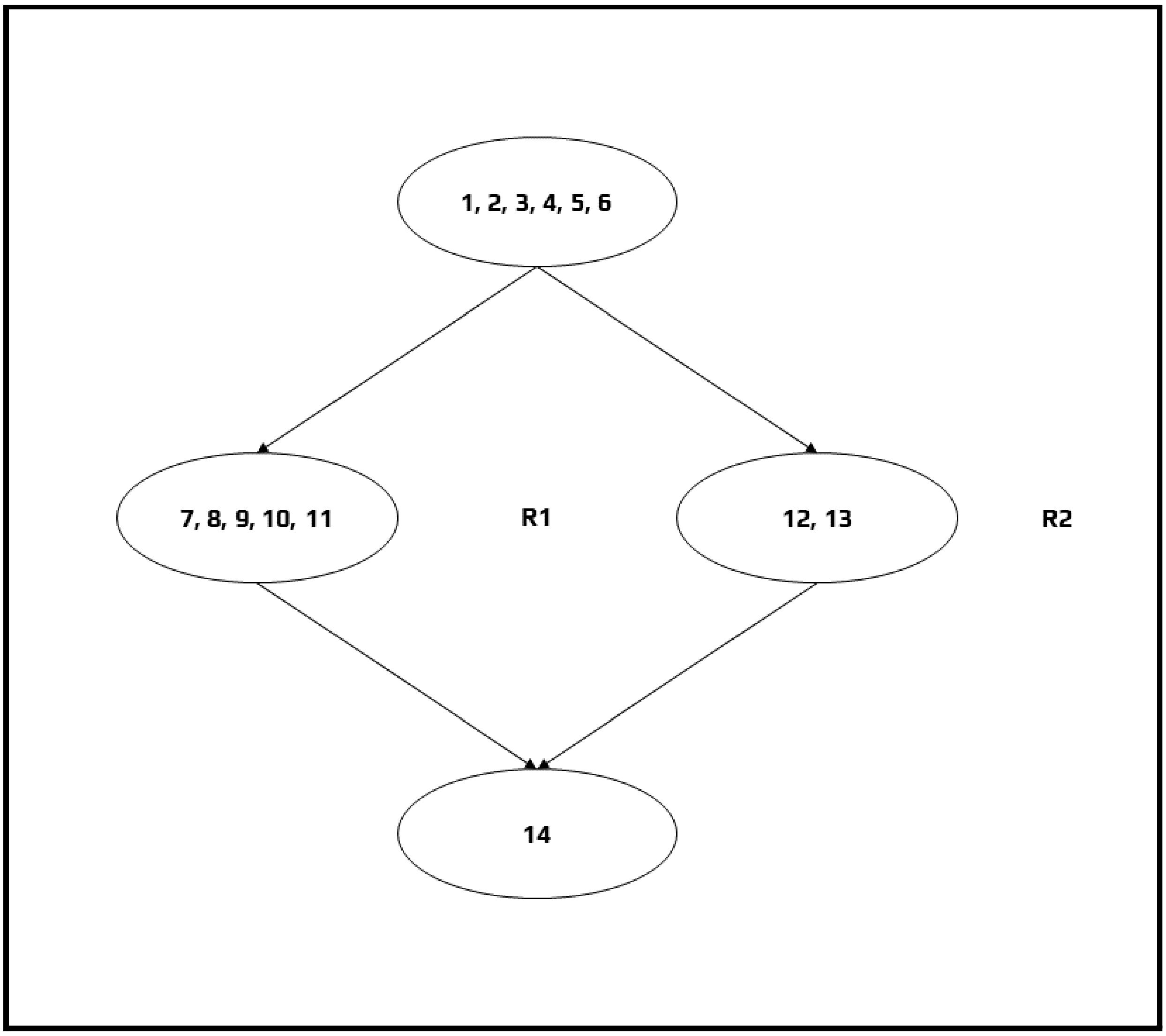
B.1 Question of Curiosity:

(Students are expected to understand the selected topic of basic path testing and complete the following)

Function

```
Function submit()            //login function//
1.  {
2.      $username=$_POST['login_username'];
3.      $password=$_POST['login_password'];
4.      $query = "SELECT * from users where UserName='$username' AND
         Password='$password'";
5.      $result = mysqli_query($con,$query)or die(mysql_error());
6.      if(mysqli_num_rows($result) > 0)
7.      {
8.          $row = mysqli_fetch_assoc($result);
9.          $_SESSION['user']=$row['UserName'];
10.           header("Location: index.php?login=" . "Successfully Logged In");
11.      }
12.      else
13.        echo "Incorrect username or password";
14.  }
```

Flow Graph



Independent Paths

Path 1: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 14
Path 2: 1 → 2 → 3 → 4 → 5 → 6 → 12 → 13 → 14

Cyclomatic Complexity

Method 1

V(G) = e - n + 2

In the above control flow graph, where, e = 4 and n = 4

Therefore, Cyclomatic Complexity

V(G) = 4 – 4 + 2

= 2

Method 2

V(G) = P + 1

In the above control flow graph, where, P = 1

Therefore, Cyclomatic Complexity

V(G) = 1 + 1

= 2

Method 3

V(G) = Number of Regions
In the above control flow graph, there are 2 regions

Test Case ID	Input Number	Output	Independent Path Covered
1	1	Login	1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 14
2	0	No Login	1 → 2 → 3 → 4 → 5 → 6 → 12 → 13 → 14

B.2 Questions

1. Differentiate between white and black box testing.

Ans:

Black Box Testing	White Box Testing
It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester knows the internal structure of the code or the program of the software.
It is mostly done by software testers.	It is mostly done by software developers.
No knowledge of implementation is needed.	Knowledge of implementation is required.

It can be referred to as outer or external software testing.	It is the inner or the internal software testing.
It is a functional test of the software.	It is a structural test of the software.
This testing can be initiated based on the requirement specifications document.	This type of testing of software is started after a detailed design document.
No knowledge of programming is required.	It is mandatory to know to program.
It is the behaviour testing of the software.	It is the logic testing of the software.
It applies to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.
It is also called closed testing.	It is also called clear box testing.
It is the least time-consuming.	It is most time-consuming.
It is not suitable or preferred for algorithm testing.	It is suitable for algorithm testing.
Can be done by trial and error ways and methods.	Data domains along with inner or internal boundaries can be better tested.
Example: search something on google by using keywords	Example: by input to check and verify loops
Types of Black Box Testing: ● A. Functional Testing ● B. Non-functional testing ● C. Regression Testing	Types of White Box Testing: ● A. Path Testing ● B. Loop Testing ● C. Condition testing

2. Differentiate between functional and non-functional testing.

Ans:

Functional Testing	**Non-functional Testing**
It verifies the operations and actions of an application.	It verifies the behaviour of an application.
It is based on the requirements of the customer.	It is based on the expectations of the customer.
It helps to enhance the behaviour of the application.	It helps to improve the performance of the application.
Functional testing is easy to execute manually.	It is hard to execute non-functional testing manually.
It tests what the product does.	It describes how the product does.
Functional testing is based on the business requirement.	Non-functional testing is based on the performance requirement.
Examples: 1. Unit Testing 2. Smoke Testing 3. Integration Testing 4. Regression Testing	Examples: 1. Performance Testing 2. Load Testing 3. Stress Testing 4. Scalability Testing

B.3 Conclusion:

(Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation)

We performed White Box Testing (Basic Path Testing) on a function of Online Bookstore and developed the test cases for the same.