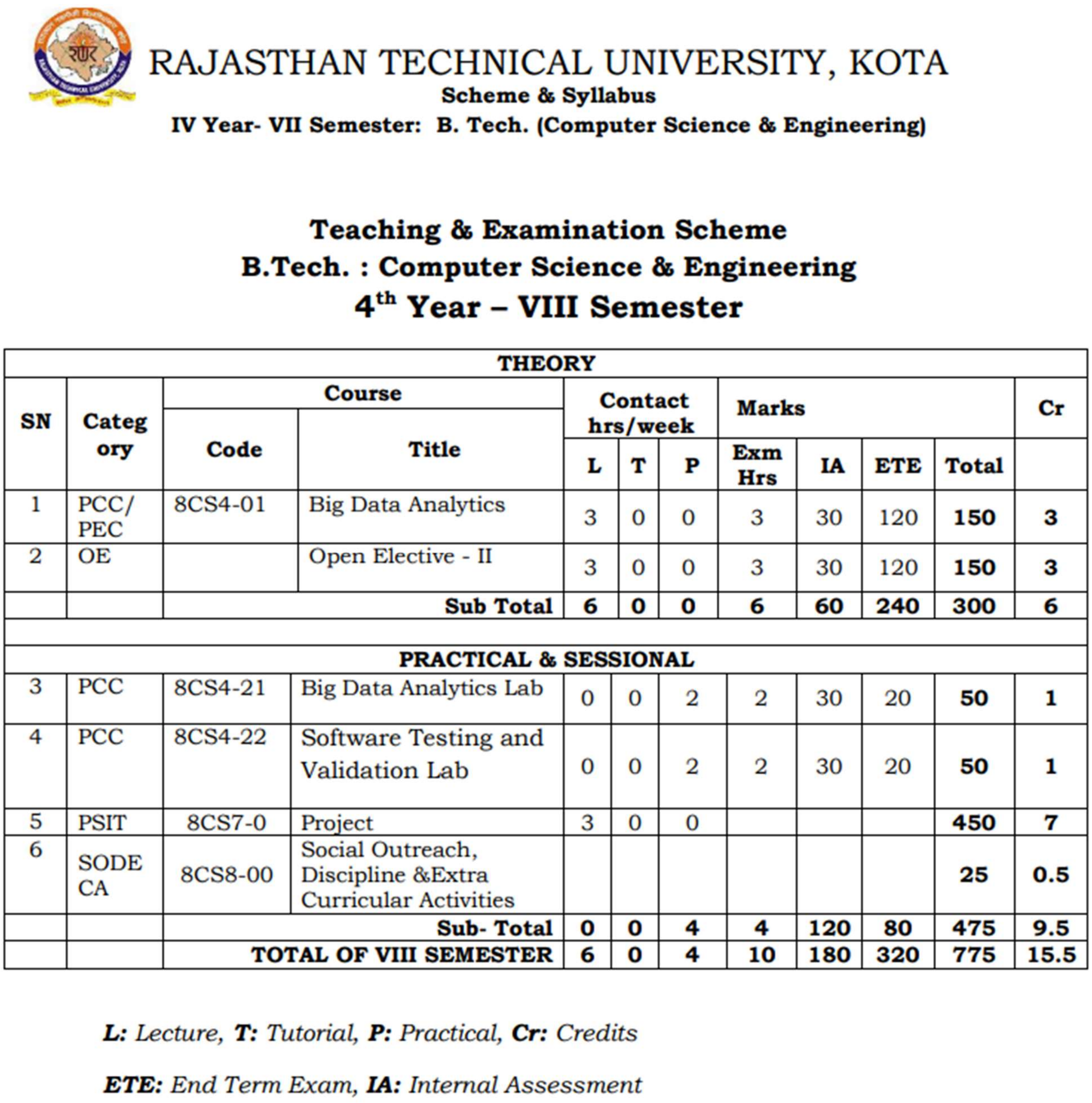
Practices for analysis, evaluation, and perception of software process, as well as learning for strong attention to small details for accurately identifying all problems and bugs quickly and effectively. Communication process to collaborate with developers or designers and precisely describe bugs to be eliminated.

Software Testing and Validation

Lab Manual

Dr Paras Kothari



RAJASTHAN TECHNICAL UNIVERSITY, KOTA

Scheme & Syllabus

IV Year- VII Semester: B. Tech. (Computer Science & Engineering)

Teaching & Examination Scheme
B.Tech. : Computer Science & Engineering
4th Year – VIII Semester

THEORY											
SN	**Category**	**Course**		**Contact hrs/week**			**Marks**				**Cr**
		Code	**Title**	**L**	**T**	**P**	**Exm Hrs**	**IA**	**ETE**	**Total**	
1	PCC/ PEC	8CS4-01	Big Data Analytics	3	0	0	3	30	120	**150**	**3**
2	OE		Open Elective - II	3	0	0	3	30	120	**150**	**3**
			Sub Total	**6**	**0**	**0**	**6**	**60**	**240**	**300**	**6**
PRACTICAL & SESSIONAL											
3	PCC	8CS4-21	Big Data Analytics Lab	0	0	2	2	30	20	**50**	**1**
4	PCC	8CS4-22	Software Testing and Validation Lab	0	0	2	2	30	20	**50**	**1**
5	PSIT	8CS7-0	Project	3	0	0				**450**	**7**
6	SODE CA	8CS8-00	Social Outreach, Discipline &Extra Curricular Activities							**25**	**0.5**
			Sub- Total	**0**	**0**	**4**	**4**	**120**	**80**	**475**	**9.5**
			TOTAL OF VIII SEMESTER	**6**	**0**	**4**	**10**	**180**	**320**	**775**	**15.5**

L: *Lecture,* ***T:*** *Tutorial,* ***P:*** *Practical,* ***Cr:*** *Credits*

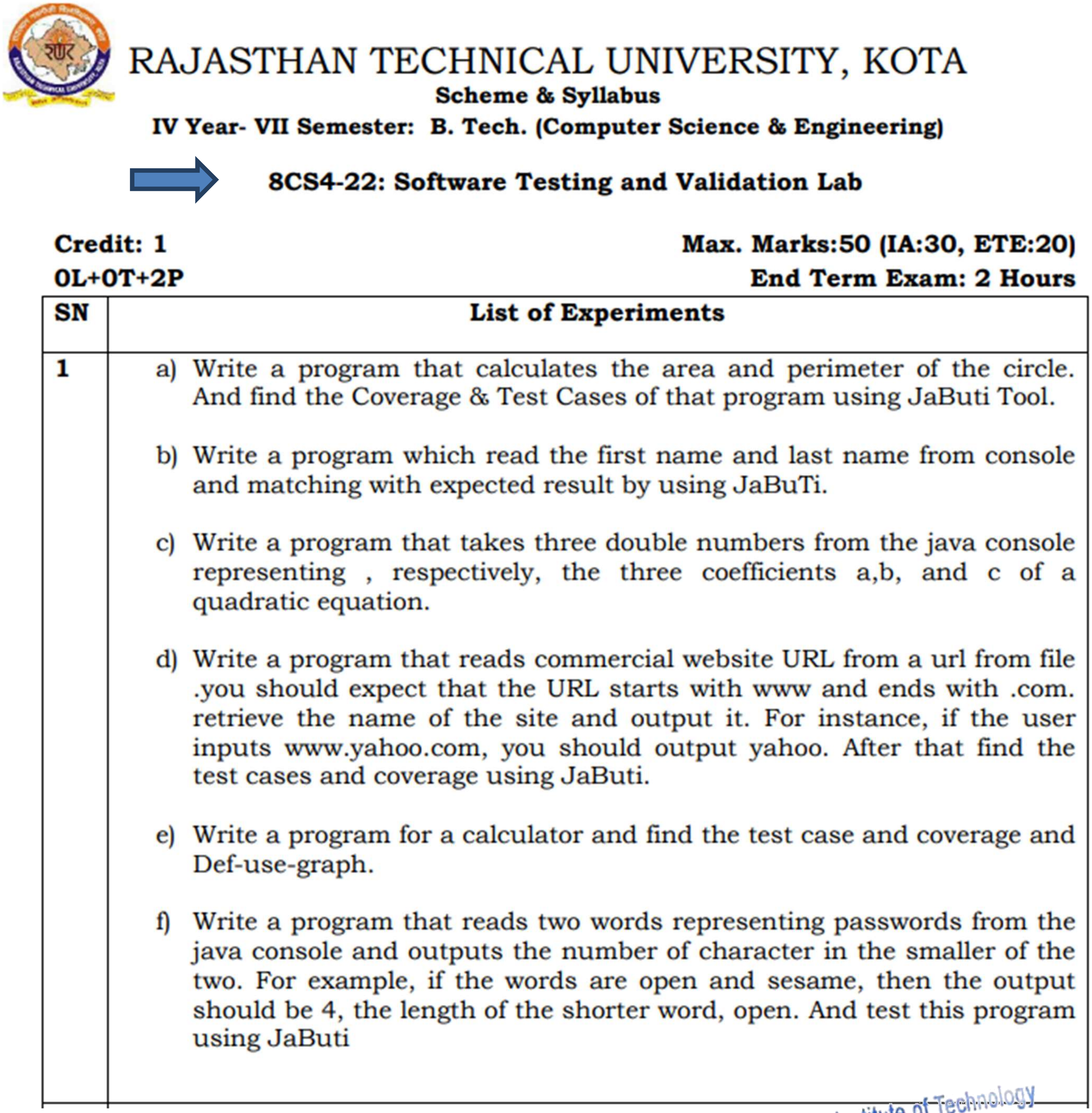
ETE: *End Term Exam,* ***IA:*** *Internal Assessment*

For Techno India NJR Institute of Technology

पंकज पोरवाल

Dr. Pankaj Kumar Porwal

(Principal)



RAJASTHAN TECHNICAL UNIVERSITY, KOTA

Scheme & Syllabus

IV Year- VII Semester: B. Tech. (Computer Science & Engineering)

8CS4-22: Software Testing and Validation Lab

Credit: 1 **Max. Marks:50 (IA:30, ETE:20)**

0L+0T+2P **End Term Exam: 2 Hours**

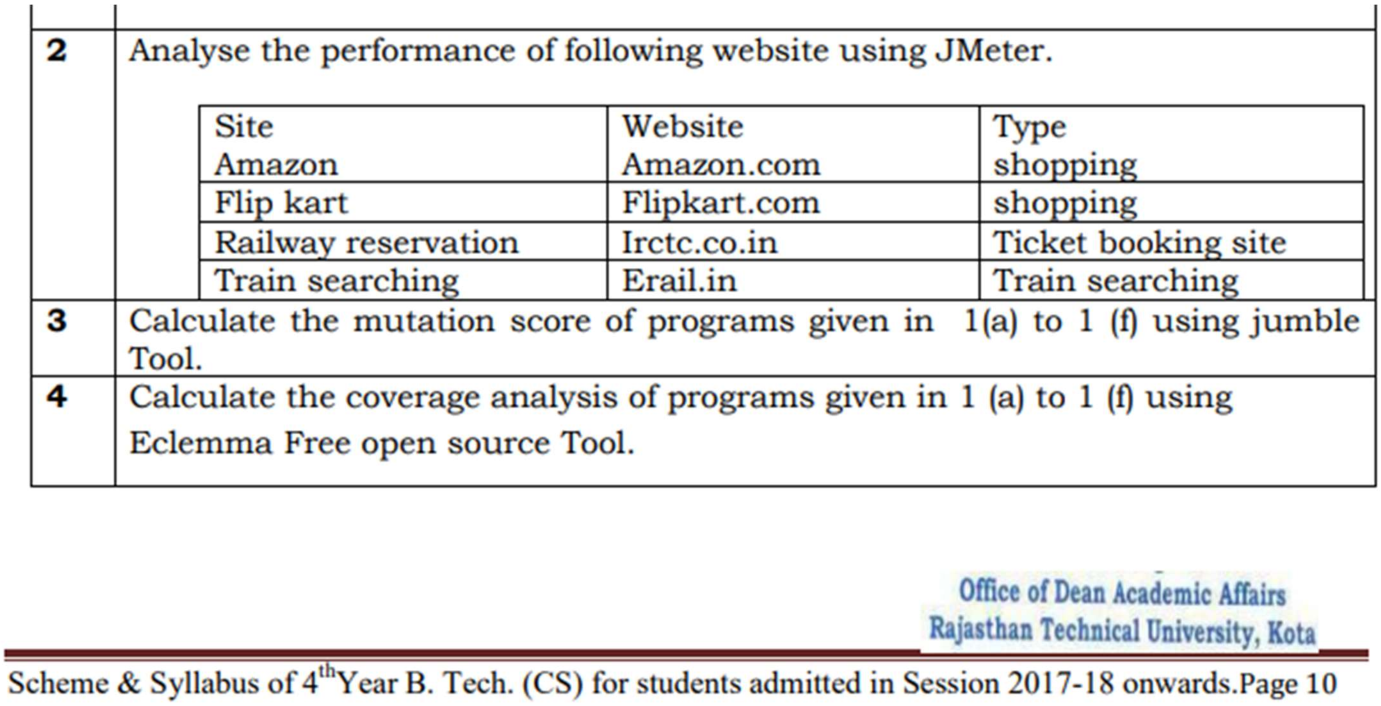
SN	List of Experiments
1	a) Write a program that calculates the area and perimeter of the circle. And find the Coverage & Test Cases of that program using JaButi Tool. b) Write a program which read the first name and last name from console and matching with expected result by using JaBuTi. c) Write a program that takes three double numbers from the java console representing , respectively, the three coefficients a,b, and c of a quadratic equation. d) Write a program that reads commercial website URL from a url from file .you should expect that the URL starts with www and ends with .com. retrieve the name of the site and output it. For instance, if the user inputs www.yahoo.com, you should output yahoo. After that find the test cases and coverage using JaButi. e) Write a program for a calculator and find the test case and coverage and Def-use-graph. f) Write a program that reads two words representing passwords from the java console and outputs the number of character in the smaller of the two. For example, if the words are open and sesame, then the output should be 4, the length of the shorter word, open. And test this program using JaButi

For Techno India NJR Institute of Technology

पंकज पोरवाल

Dr. Pankaj Kumar Porwal

(Principal)



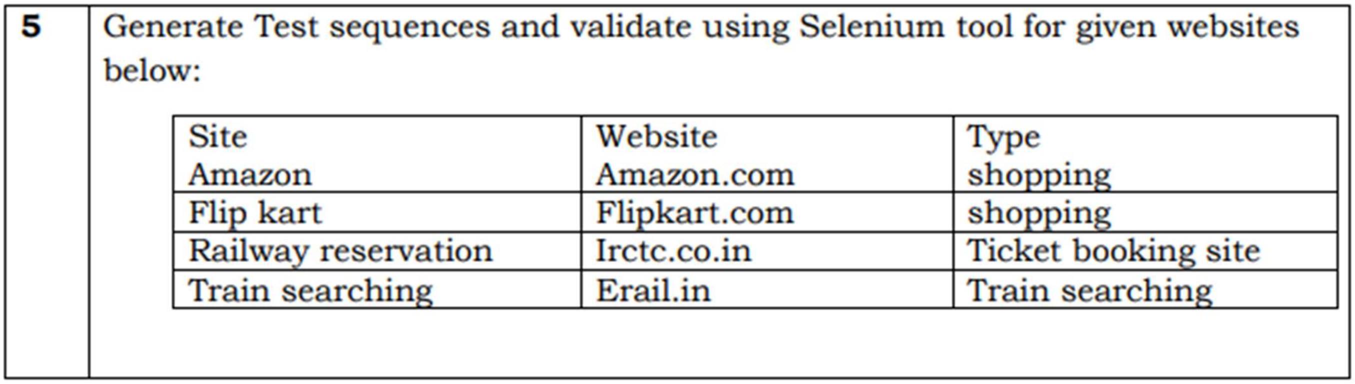
2 Analyse the performance of following website using JMeter.

Site	Website	Type
Amazon	Amazon.com	shopping
Flip kart	Flipkart.com	shopping
Railway reservation	Irctc.co.in	Ticket booking site
Train searching	Erail.in	Train searching

3 Calculate the mutation score of programs given in 1(a) to 1 (f) using jumble Tool.

4 Calculate the coverage analysis of programs given in 1 (a) to 1 (f) using Eclemma Free open source Tool.

Office of Dean Academic Affairs
Rajasthan Technical University, Kota



5 Generate Test sequences and validate using Selenium tool for given websites below:

Site	Website	Type
Amazon	Amazon.com	shopping
Flip kart	Flipkart.com	shopping
Railway reservation	Irctc.co.in	Ticket booking site
Train searching	Erail.in	Train searching

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

Software Testing and Validation Lab

EXPERIMENT 1:

AIM: Consider an automated banking application. The user can dial the bank from a personal computer, provide a six-digit password, and follow with a series of keyword commands that activate the banking function. The software for the application accepts data in the following form:

Area Code	Blank or three-digit number
Prefix	Three-digit number, not beginning with 0 or 1
Suffix	Four-digit number
Password	Six-character alphanumeric
Commands	"Check status", "Deposit", "Withdrawal"

Design ad-hoc test cases to test the system.

DESCRIPTION:-

Ad-hoc Testing:

Ad-hoc testing is an informal and improvisational approach to assessing the viability of a product. An ad-hoc test is usually only conducted once unless a defect is found. Commonly used in software development, ad hoc testing is performed without a plan of action and any actions taken are not typically documented. Testers may not have detailed knowledge of product requirements. Ad hoc testing is also referred to as random testing and monkey testing.

Because the approach is non-methodical, ad hoc testing can miss flaws that would be found in a more structured testing system. However, the lack of formal requirements also means that obvious flaws can be attended to more quickly than if they had to be approached in a more systematic fashion.

TEST CASES:

TestCase ID	Test Step	Description	Test Data	Expected result	Actual result	Status Pass/Fai l)
TC01	validation of area code	enter the 3 digit area code if non- local and Blank if local.				
TC02	validation of area code	enter the 3 digit area code if non- local and Blank if local.				
TC03	validation of area code	enter the 3 digit area code if non- local and Blank if local.				
TC04	validation of area code	enter the 3 digit area code if non- local and Blank if local.				
TC05	Validation of Prefix	enter a 3-digit number prefix. It should not begin with 0 (or) 1.				

TC06	Validation of Prefix	enter a 3-digit number prefix.It should not begin with 0 (or) 1.			6	
TC07	Validation of Prefix	enter a 3-digit number prefix.It should not begin with 0 (or) 1.				
TC08	Validation of Suffix	enter a 4-digit number suffix.				
TC09	Validation of Suffix	enter a 4-digit number suffix.				
TC10	Validation of Password	enter a Six-character alphanumeric password.				
TC11	Validation of Password	enter a Six-character alphanumeric password.				
TC12	Validation of Password	enter a Six-character alphanumeric password.				

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

TC13	Validation of commands	operate the commands "Check status","Deposit", "Withdrawl".				
TC14	Validation of commands	Operate the commands "Check status","Deposit", "Withdrawl".				
TC15	Validation of commands	operate the commands "Check status","Deposit", "Withdrawl".				

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

VIVA VOCE QUESTIONS:

1. What are the Myths of Software Testing?

2. What are the different goals of Software Testing?

3. Is Exhaustive software testing possible?

4. What is the difference between Verification & Validation?

5. What is a Failure?

6. What is a Bug?

7. What is a Error?

8. What is Testware?

9. What is Incident?

10 .What is Test Oracle?

For Techno India NJR Institute of Technology

पंकज पोरवाल

Dr. Pankaj Kumar Porwal

(Principal)

EXPERIMENT-2

AIM: Consider an automated banking application. The user can dial the bank from a personal computer, provide a six-digit password, and follow with a series of keyword commands that activate the banking function. The software for the application accepts data in the following form:

Area Code	Blank or three-digit number
Prefix	Three-digit number, not beginning with 0 or 1
Suffix	Four-digit number
Password	Six-character alphanumeric
Commands	"Check status", "Deposit", "Withdrawal"

Design the test cases to test the system using following Black Box testing technique:

BVA, Worst BVA, Robust BVA, Robust Worst BVA

DESCRIPTION:

BOUNDARY VALUE ANALYSIS:

Boundary value analysis is the process of testing between extreme ends or boundaries between partitions of the input values. So, these extreme ends like min, min+, nom, max, max- values are called boundary values and the testing is called "boundary testing".

The basic idea in boundary value testing is to select input variable values at their:

1. Minimum

2. Just above the minimum

3. A nominal value

4. Just below the maximum

5. Maximum

ROBUST BVA:

Robustness testing can be seen as an extension of Boundary Value Analysis. The idea behind Robustness testing is to test for clean and dirty test cases. By clean I mean input variables that lie in the legitimate input range. By dirty I mean using input variables that fall just outside this input domain. In addition to the aforementioned 5 testing values (min, min+, nom, max-, max) we use two more values for each variable (min-, max+), which are designed to fall just outside of the input range.

WORST BVA:

Boundary Value analysis uses the critical fault assumption and therefore only tests for a single variable at a time assuming its extreme values. By disregarding this assumption we are able to test the outcome if more than one variable were to assume its extreme value. In an electronic circuit this is called Worst Case Analysis.

ROBUST WORST BVA:

If the function under test were to be of the greatest importance we could use a method named Robust Worst-Case testing which as the name suggests draws it attributes from Robust and Worst-Case testing. Test cases are constructed by taking the Cartesian product of the 7-tuple set defined in the Robustness testing chapter. Obviously this results in the largest set of test results we have seen so far and requires the most effort to produce

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

TEST CASES:

TEST CASES USING BOUNDARY VALUE ANALYSIS:

	Area Code	Prefix	Suffix	Password	Command
Min value	blank	200	0	a0A0a0	S/D/W
Min+ value	11	201	1	b1B1b1	SD/SW/DS
Max Value	934	999	9999	z9Z9z9	SDW
Max- value	863	998	9998	x8X8x8	SD/SW/DS
Nominal value	473	500	5000	m5M5m5	SD

Test Case ID	Area Code	Prefix	Suffix	Password	Command	Expected output
TC01						
TC02						
TC03						
TC04						
TC05						

TEST CASES USING ROBUST BVA:

	Area Code	Prefix	Suffix	Password	Command
Min- value	0	199	999	!@#$%	no input
Min value	Blank	200	1111	a0A0a0	S/D/W
Min+ value	11	201	1112	b1B1b1	SD/SW/DS
Max value	934	999	9999	z9Z9z9	SDW
Max- value	863	998	9998	x8X8x8	SD/SW/DS
Max+ value	999	1000	10000	?><{}	WDS
Nominal value	473	500	5000	m5M5m5	SD

Test CaseID	**Area Code**	**Prefix**	**Suffix**	**Password**	**Command**	**Expected output**
TC01						
TC02						
TC03						
TC04						
TC05						

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

TEST CASES USING WORST CASE TESTING:

	Area Code	Prefix	Suffix	Password	Command
Min value	blank	200	0	a0A0a0	S/D/W
Min+ value	11	201	1	b1B1b1	SD/SW/DS
Max Value	934	999	9999	z9Z9z9	SDW
Max- value	863	998	9998	x8X8x8	SD/SW/DS
Nominal value	473	500	5000	m5M5m5	SD

Test CaseID	Area Code	Prefix	Suffix	Password	Command	Expected output
TC01						
TC02						
TC03						
TC04						
TC05						

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

VIVA VOCE QUESTIONS:

1. Mention some of the static & dynamic testing techniques?
2. What is an equivalence partition?
3. What is the purpose of test completion?
4. When should testing be stopped?
5. How many no of test cases are needed in BVA?
6. How many no of testcases are needed in ROBUST BVA?
7. How many no of testcases are needed in Worst CASE Testing ?
8. Faults found ,should be originally documented by whom?
9. Why does BVA provide good testcases?
10. What is test coverage?

EXPERIMENT-3

AIM: Consider an application that is required to validate a number according to the following simple rules

- A number can start with an optional sign.
- The optional sign can be followed by any number of digits.
- The digits can be optionally followed by a decimal point, represented by a period.
- If there is a decimal point, then there should be two digits after the decimal.
- Any number-whether or not it has a decimal point, should be terminated a blank.
- A number can start with an optional sign.
- The optional sign can be followed by any number of digits.
- The digits can be optionally followed by a decimal point, represented by a period.
- If there is a decimal point, then there should be two digits after the decimal.
- Any number-whether or not it has a decimal point, should be terminated a blank. Generate test cases to test valid and invalid numbers.

DESCRIPTION:

DECISION TABLE BASED TESTING:

A decision table is an excellent tool to use in both testing and requirements management. Essentially it is **a structured exercise to formulate requirements when dealing with complex business rules**. Decision tables are used to model complicated logic.

In a decision table, **conditions are usually expressed as true (T) or false (F)**. Each column in the table corresponds to a rule in the business logic that describes the unique combination of circumstances that will result in the actions.

Decision tables can be used in all situations where the outcome depends on the combinations of different choices, and that is usually very often. In many systems there are tons of business rules where decision tables add a lot of value.

Steps for Constructing Decision Table and generating Test cases:

1. **Analyze the requirement and create the first column**
2. **Add Columns**
3. **Reduce the Table**
4. **Determine Actions**
5. **Write Test Cases**

DECISION TABLE:

		Rule 1	Rule 2	Rule 3	Rule 4
Condition Stub	C1:Number can start with optional sign	T	T	T	T
	C2:Optional sign can be followed by any num of digits	T		T	T
	C3:Digits can be optionally followed by a decimal point	T	T		T
	C4:There should be 2 digits after a decimal point		F		T
	C5:Number should be terminated with a blank	F	T	T	T
Action Stub	A1:Valid number			*	*
	A2:invalid number	*	*		

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

TEST CASES:

Test case ID	**optional sign**	**Digits**	**decimal point**	**no.of decimal digits**	**termination blank**	**expected output**
TC01						
TC02						
TC03						
TC04						
TC05						

VIVA VOCE QUESTIONS:

1. What is Decision Table Based Testing?
2. What is Error Guessing?
3. What are the different notations used in Cause Effect Graph?
4. What is strong equivalence class?
5. What is weak equivalence class?
6. How do you generate the test cases in Decision Table Based Testing?

For Techno India NJR Institute of Technology

पंकज पोरवाल

Dr. Pankaj Kumar Porwal
(Principal)

EXPERIMENT-4

AIM: Generate test cases using Black box testing technique to Calculate Standard Deduction on Taxable Income. The standard deduction is higher for tax payers who are 65 or older or blind. Use the method given below to calculate tax.

1. The first factor that determines the standard deduction is the filing status. The basic standard deduction for the various filing status are:

Single	$4,750
Married, filing a joint return	$9,500
Married, filing a separate return	$7,000

2. If a married couple is filing separate returns and one spouse is not taking standard Deduction, the other spouse also is not eligible for standard deduction.
3. An additional $1,000 is allowed as standard deduction, if either the filer is 65 yrs or the spouse is 65 yrs or older (the latter case applicable when the filing status is “Married” and filing “joint”).
4. An additional $1,000 is allowed as standard deduction, if either the filer is blind or the spouse is blind (the latter case applicable when the filing status is
“married” and filing “joint”).

DESCRIPTION:

BLACK-BOX TESTING:

` Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or working. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher level testing ,but can also dominate unit testing as well.

EQUIVALENCE CLASS:

Equivalence Partitioning also called as equivalence class partitioning. It is abbreviated as ECP. It is a software testing technique that divides the input test data of the application under test into each partition at least once of equivalent data from which test cases can be derived. An advantage of this approach is it reduces the time required for performing testing of a software due to less number of test cases.

EQUIVALENCE CLASSES:

C1= Status of the {Single||Married}

C2=Age of the {>=65&<65}

C3=Eye sight {Blind || No blind}

C4=Class {separate || joint}

TEST CASES

TEST CASE ID	**Class Covered**	**Status of the Filer**	**Age of the Filer**	**Eye sight**	**Class**	**Expected output**	**ActualOutput (Pass /Fail)**
SD1							
SD2							
SD3							
SD4							
SD5							
SD6							
SD7							
SD8							
SD9							
SD10							

For Techno India NJR Institute of Technology

पंकज पोरवाल

Dr. Pankaj Kumar Porwal

(Principal)

VIVA VOCE QUESTIONS:

1. What is a V-Model?
2. What is Strong Equivalence Class?
3. What is Weak Equivalence Class?
4. What are Semi Random Test Cases?
5. What is Black Box Testing?
6. Why we split testing into different stages?
7. What is Fault Masking?
8. What is the Difference between STLC & SDLC?
9. What is the difference between Test Scenarios & Test Cases & Test Script?
10. What are the Test Delivarables?

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

EXPERIMENT-5

AIM: Consider the following program segment:

```
1. int max (int i, int j, int k)
2. {
3.int max;
4.if (i>j) then
5.if (i>k) then
6..max=i;
7.else max=k;
8.else if (j > k)
9. max=j
10.else max=k
11.return (max);
12.}
```

a) Draw the control flow graph for this program segment

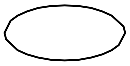
b) Determine the cyclomatic complexity for this program

c) Determine the independent paths

DESCRIPTION:-

CONTROL FLOW GRAPH:

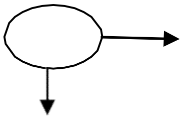
- The control flow graph is a graphical representation of a program's control structure. It uses the elements named process blocks, decisions, and junctions.
- The flow graph is similar to the earlier flowchart, with which it is not to be confused.
- The nodes are represented in the following way



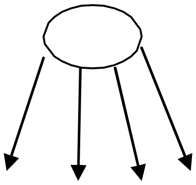
- **Flow Graph Elements:** A flow graph contains four different types of elements (1) Decisions (2) Junctions (3) Case Statements.

1. **Decisions:**
    - A decision is a program point at which the control flow can diverge.

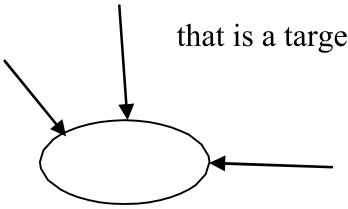
- Machine language conditional branch and conditional skip instructions are examples of decisions.
- Most of the decisions are two-way but some are three way branches in control flow.



2. **Case Statements:**
    - A case statement is a multi-way branch or decisions.
    - Examples of case statement are a jump table in assembly language, and the PASCAL case statement.
    - From the point of view of test design, there are no differences between Decisions and Case Statements



3. **Junctions:**
    - A junction is a point in the program where the control flow can merge.
    - Examples of junctions are: the target of a jump or skip instruction in ALP, a label that is a target of GOTO.



CYCLOMATIC COMPLEXITY:

Cyclomatic complexity is a source code complexity measurement that is being correlated to a number of coding errors. It is calculated by developing a Control Flow Graph of the code that measures the number of linearly-independent paths through a program module.

In other words, the **cyclomatic complexity** metric is based on the number of decisions in a program. It is important to testers because it provides an indication of the amount of testing (including reviews) necessary to practically avoid defects.

In other words, areas of code identified as more complex are candidates for reviews and additional dynamic tests. While there are many ways to calculate cyclomatic complexity, the easiest way is to sum the number of binary decision statements (e.g. if, while, for, etc.) and add 1 to it. A more formal definition regarding the calculation rules is provided in the glossary.

Cyclomatic complexity is measured in three ways:

I. Cyclomatic complexity = E - N + P

Where, E = number of edges in the flow graph.
N = number of nodes in the flow graph.
P = number of Procedures in the flow graph

II. Cyclomatic complexity = D+1

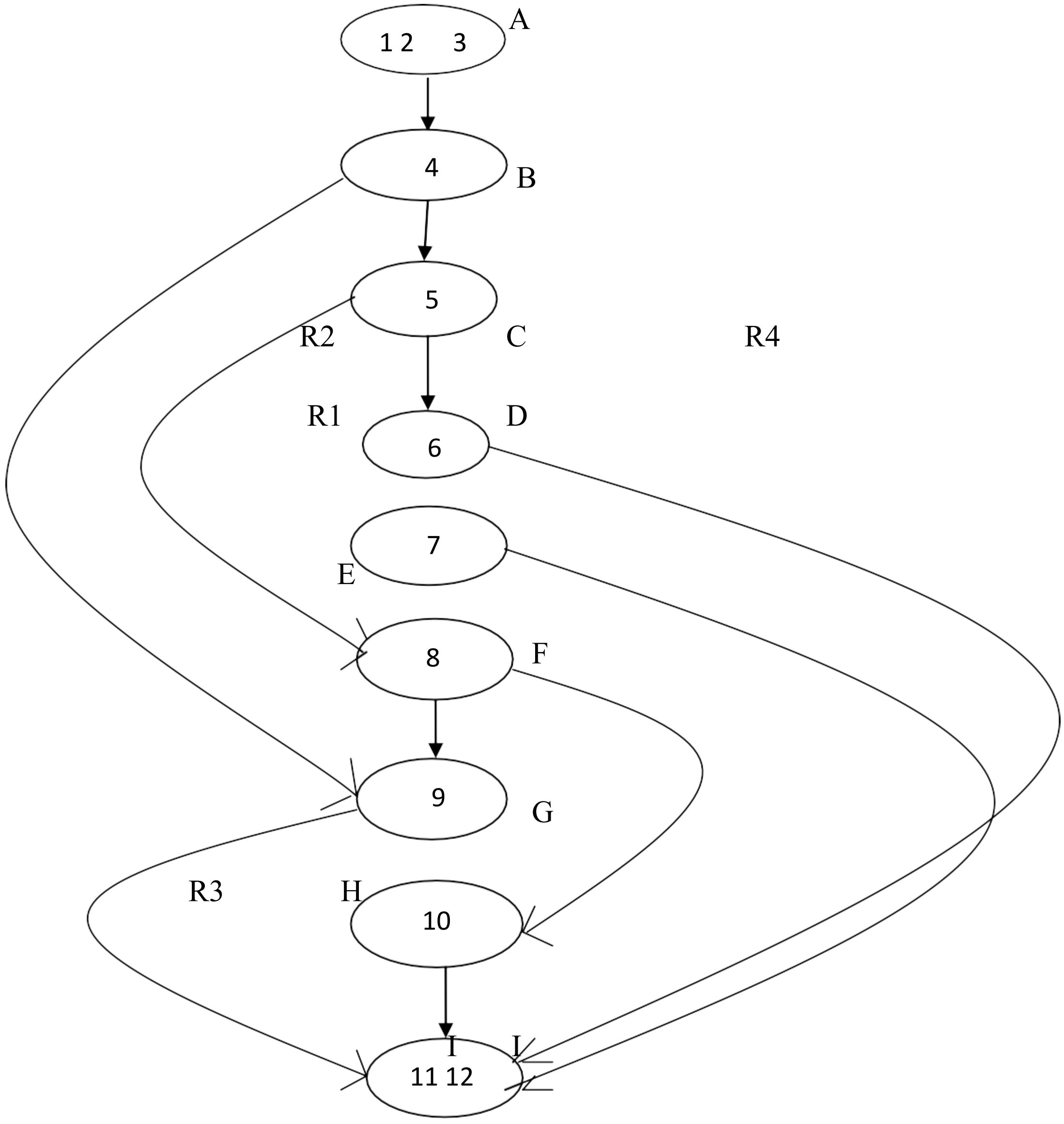
Where, D= number of decision statements in the flow graph

III. Cyclomatic complexity = number of regions in graph G.

INDEPENDENT PATHS:

The number of different paths from the stating node of the flow graph to the ending node of the flow graph. A graph may have 1 or more independent paths.

A) CONTROL FLOW GRAPH:



B) CYCLOMETRIC COMPLEXITY:

V(G)=e-n+2P

=

V(G)=d+1

=

V(G)= no.of regions in graph G

=

C) INDEPENDENT PATHS:

VIVA VOCE QUESTIONS:

1) What is Control Flow Graph?

2) What is Cyclomatic Complexity?

3) How do you determine no of Independent paths in a program segment?

4) What is White Box Testing?

5) What is meant by Decision Node?

6) What is the difference between Segment & Path Segment?

7) What is the Junction Node?

EXPERIMENT-6

AIM: source code of simple insertion sort implementation using arrays in ascending order in c programming language

PROGRAM:

```
#include<stdio.h>
Int main()
{
IntI,j,s,temp,a[20];
Printf("Enter total elements");
Scanf("%d",&s);
Printf("Enter %d elements",s);
For(i=0;i<s;i++)
Scanf("%d",a[i]);
For(i=0;i<s;i++)
{
Temp=a[i];
J=i-1;
While((temp<a[j])&&(j>=0))
{
a[j+1]=a[j];
j=j-1;
}
a[j+1]=temp;
}
Printf("after sorting");
For(i=0;i<s;i++)
Printf("%d",a[i]);
return 0;
}
```

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

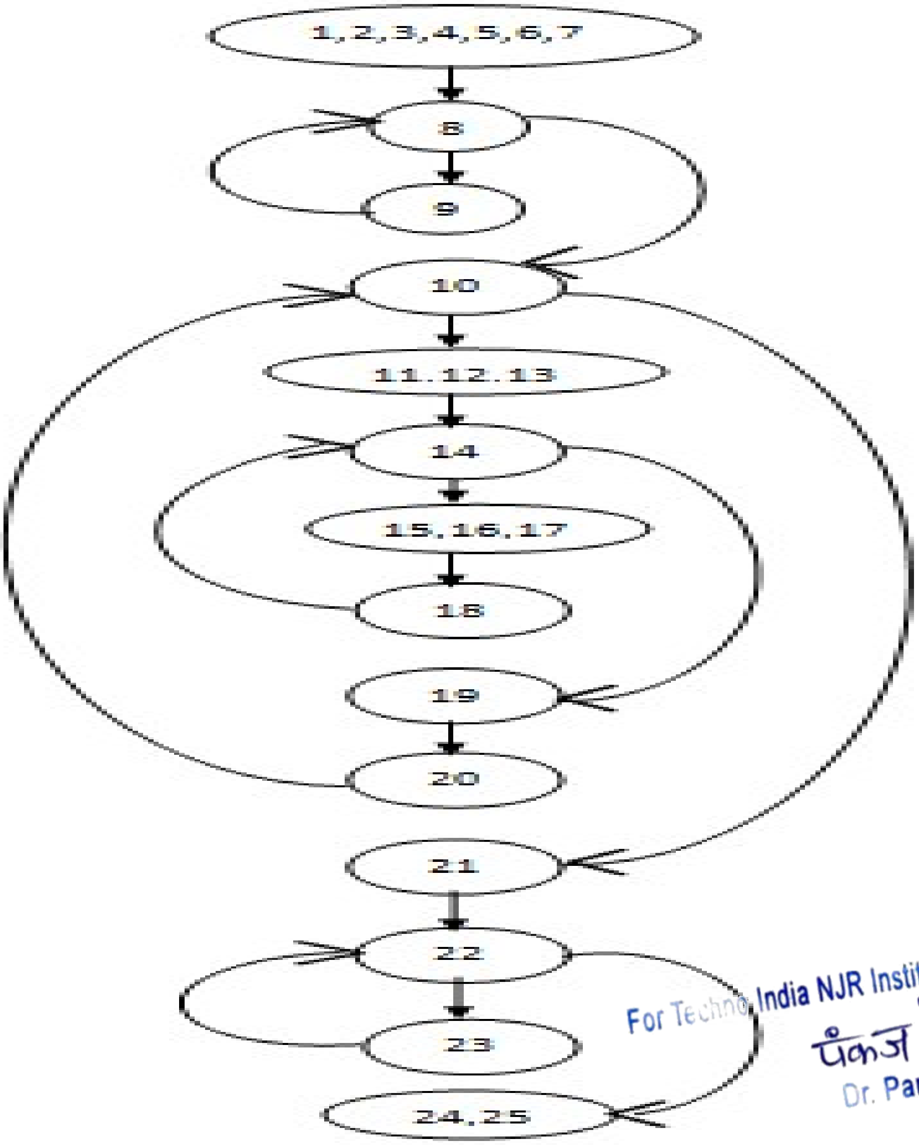
a) Draw control flow graph

b) Determine the cyclomatic complexity

c) Determine independent paths.

d) Design test cases for this program segment.

A) **CONTROL FLOW GRAPH:**



B) CYCLOMATIC COMPLEXITY:

a. V(g) = e-n+2p =

b. V(g) = d+1 =

c. No „of regions: v(g)=

C) INDEPENDENT PATHS :

D) TEST CASES :

Test cases	Input	Expected Output	Independent Path
TC01			
TC02			
TC03			
TC04			

VIVA VOCE QUESTIONS:

1) What is the need of White Box Testing?

2) What is the difference between White Box & Black Box Testing?

3) What is meant by Basis Path Testing?

4) Discuss the applications of Path testing?

5) What is Graph Matrix?

6) What is static testing?

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

EXPERIMENT-7

AIM: Consider a system having an FSM for a stack having the following states and Transitions.

STATES:

Initial: Before creation

Empty: Number of elements =0

Holding: Number of elements > 0, but less than the maximum capacity.

Full: Number of elements = maximum

Final: After destruction

Initial to Empty: Create

Empty to Holding, Empty to Full, Holding to Holding, Holding to Full: add

Empty to Final, Full to Final, Holding to Final: Destroy

Holding to Empty, Full to Holding, Full to Empty: Delete

Design test case for this FSM using state table- based testing.

DESCRIPTION:

State Transition testing, a black box testing technique, in which outputs are triggered by changes to the input conditions or changes to 'state' of the system. In other words, tests are designed to execute valid and invalid state transitions.

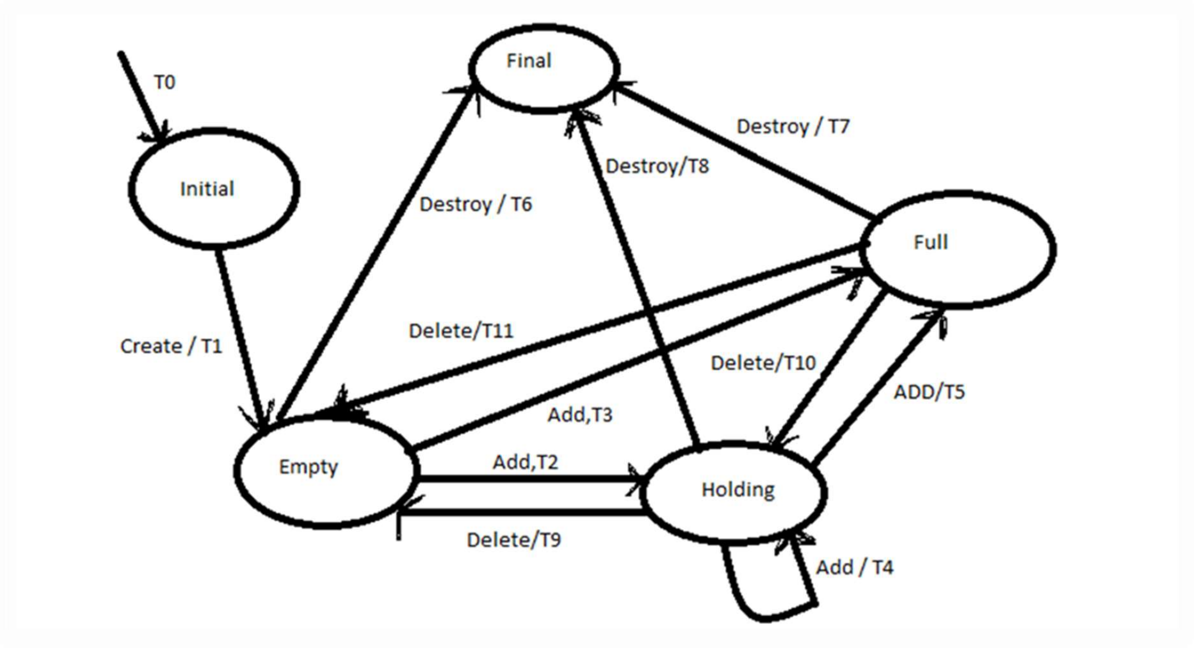
DERIVING TEST CASES:

- Understand the various state and transition and mark each valid and invalid state
- Defining a sequence of an event that leads to an allowed test ending state
- Each one of those visited state and traversed transition should be noted down
- Steps 2 and 3 should be repeated until all states have been visited and all transitions traversed

- For test cases to have a good coverage, actual input values and the actual output values have to be generated

ADVANTAGES:

- Allows testers to familiarize with the software design and enables them to design tests effectively.
- It also enables testers to cover the unplanned or invalid states.



STATE TABLE:

State / Event	Create	Add	Delete	Destroy
Initial	Empty / T1	Initial / T0	Initial / T0	Initial / T0
Empty	Empty / T1	Holding / T2 Full / T3	Empty / T1	Final / T6

Holding	Holding / T2	Holding / T4 Full / T5	Empty / T9	Final / T8
Full	Full / T3	Full / T3	Holding / T10 Empty / T11	Final / T7
Final	Final / T6	Final / T6	Final / T6	Final / T6

TEST CASES:

	Input		**Expected Output**	
Test Case ID	**State**	**Event**	**O/P**	**Next State**
TC01				
TC02				
TC03				
TC04				
TC05				
TC06				
TC07				
TC08				
TC09				
TC10				

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

TC11				
TC12				
TC13				
TC14				
TC15				
TC16				
TC17				
TC18				
TC19				
TC20				
TC21				
TC22				
TC23				

VIVA VOCE QUESTIONS:

1. What is State Table Based Testing?
2. What is FSM?
3. What is meant by state transition?
4. What is meant by error guessing?
5. Each row of state table corresponding to what?
6. Each column of state table corresponding to what?

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

EXPERIMENT-8

AIM : Given the following fragments of code, how many tests are required for 100% decision coverage ? Give the test cases.

PROGRAM:

```
If width > length then
        biggest_dimension=width
        If height > width
                biggest_dimension = height
        end
else
        biggest_dimension = length
        if height > length
                biggest_dimension = height
        end_if
end_if
```

DESCRIPTION:

Branch coverage is also known as Decision coverage or all-edges coverage.It covers both the true and false conditions unlikely the statement coverage.A branch is the outcome of a decision, so branch coverage simply measures which decision outcomes have been tested. This sounds great because it takes a more in-depth view of the source code than simple statement coverage.

A decision is an IF statement, a loop control statement (e.g. DO-WHILE or REPEAT-UNTIL), or a CASE statement, where there are two or more outcomes from the statement. With an IF statement, the exit can either be TRUE or FALSE, depending on the value of the logical condition that comes after IF.

ADVANTAGES OF DECISION COVERAGE:

- To validate that all the branches in the code are reached
- To ensure that no branches lead to any abnormality of the program‟soperation
- It eliminates problems that occur with statement coverage testing

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

DISADVANTAGES OF DECISION COVERAGE:

- This metric ignores branches within Boolean expressions which occur due to short-circuit operators.

$$\text{Decision Coverage} = \frac{\text{Number of decision outcomes exercised}}{\text{Total number of decision outcomes}} \times 100$$

TEST CASES:

Test case id	Input	Expected output	Actual Output
TC01			
TC02			
TC03			
TC04			

VIVA VOCE QUESTIONS:

1) What is meant by Branch Coverage?

2) What is meant by Condition Coverage?

3) What is meant by Decision/Condition Coverage?

4) What is the criteria for Logic Coverage?

EXPERIMENT-9

AIM : To give the following code ., how much minimum number of test cases is required for full statement and branch coverage ?

PROGRAM:

```
Read p

Read q

If p+q>100

Then print “large: endif

If p>50

Then print “p large” endif
```

DESCRIPTION:

STATEMENT COVERAGE:

Statement coverage is a white box testing technique, which involves the execution of all the statements at least once in the source code. It is a metric, which is used to calculate and measure the number of statements in the source code which have been executed. Using this technique, we can check what the source code is expected to do and what it should not. It can also be used to check the quality of the code and the flow of different paths in the program. The main drawback of this technique is that we cannot test the false condition in it.

Advantage of statement coverage:

- It verifies what the written code is expected to do and not to do
- It measures the quality of code written
- It checks the flow of different paths in the program and it also ensure that whether these path are tested or not.

Disadvantage of statement coverage:

- It cannot test the false conditions.
- It does not report that whether the loop reaches its termination condition.
- It does not understand the logical operators.

$$\text{Statement coverage} = \frac{\text{Number of statements Executed}}{\text{Total number of statements in the source code}} \times 100$$

Branch Coverage:

Branch coverage is also known as Decision coverage or all-edges coverage.It covers both the true and false conditions unlikely the statement coverage.A branch is the outcome of a decision, so branch coverage simply measures which decision outcomes have been tested. This sounds great because it takes a more in-depth view of the source code than simple statement coverage.

A decision is an IF statement, a loop control statement (e.g. DO-WHILE or REPEAT-UNTIL), or a CASE statement, where there are two or more outcomes from the statement. With an IF statement, the exit can either be TRUE or FALSE, depending on the value of the logical condition that comes after IF.

Advantages of decision coverage:

- To validate that all the branches in the code are reached
- To ensure that no branches lead to any abnormality of the program‟s operation
- It eliminates problems that occur with statement coverage testing

Disadvantages of decision coverage:

- This metric ignores branches within Boolean expressions which occur due to short-circuit operators.

$$\text{Decision Coverage} = \frac{\text{Number of decision outcomes exercised}}{\text{Total number of decision outcomes}} \times 100$$

TEST CASES:

i) Statement Coverage (minimum) :

Test Case ID	Input	Expected Output
TC01		

ii) Branch Coverage (minimum):

Test Case ID	Input	Expected Output
TC01		
TC02		

VIVA VOCE QUESTIONS:

1. **What is meant by statement coverage?**
2. **What is meant by multiple condition coverage?**
3. **Explain what is Test Plan? What are the information that should be covered in TestPlan ?**
4. **What is test management review and why it is important?**
5. **What is the difference between Test Matrix &Tracebility Matrix?**
6. **Mention the different Categories of defects?**
7. **What are the different types of test coverage techniques?**
8. **What is meant by an acceptance testing?**

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

EXPERIMENT-10

AIM: Consider a program to input two numbers and print them in ascending order given below. Find all du paths and identify those du-paths that are not feasible. Also find all dc paths and generate the test cases for all paths (dc paths and non dc paths).

```
#include<stdio.h>
#include<conio.h>
1. void main ()
2. {
3 int a, b, t;
4. Clrscr ();
5. Printf (“Enter first number”);
6. scanf (“%d”,&a);
7. printf(“Enter second number”);
8. scanf(“%d”,&b);
9. if (a<b){
10. t=a;
11. a=b;
12. b=t;
13. }
14. printf (“%d %d”, a, b);
15 getch ();
}
```

DESCRIPTION:

DU-PATHS:-

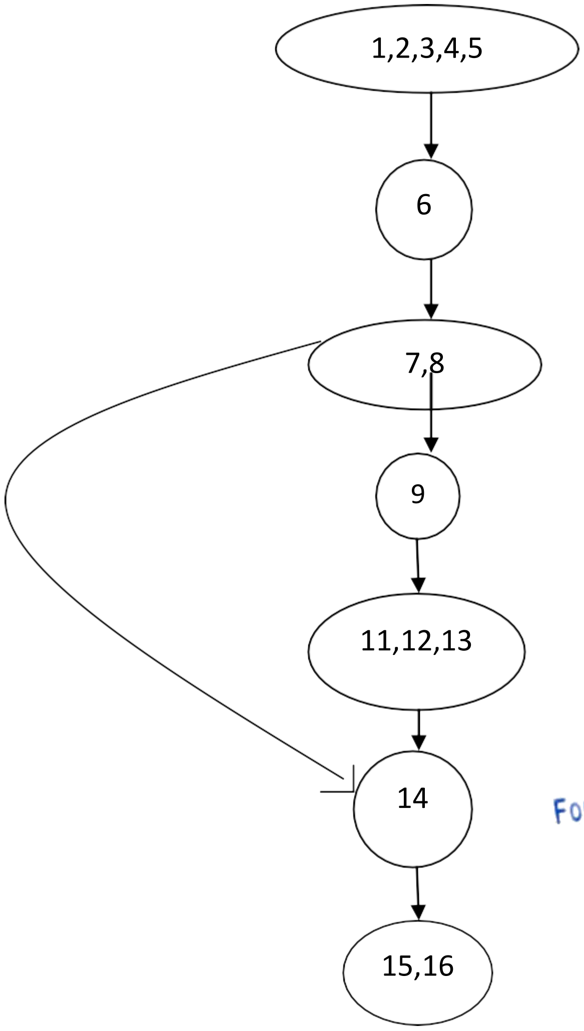
In a path segment if the variable is defined earlier and the last link has a computational use of that variable then that path is termed as a du path.

A sub-path in the flow is defined to go from a point where a variable is "defined", to a point where it is "referenced", that is, where it is "used" - whatever kind of usage it is. Such a sub-path is called a "definition-use pair" or "du-pair". The pair is made up of a "definition" of a variable and a "use" of the variable.

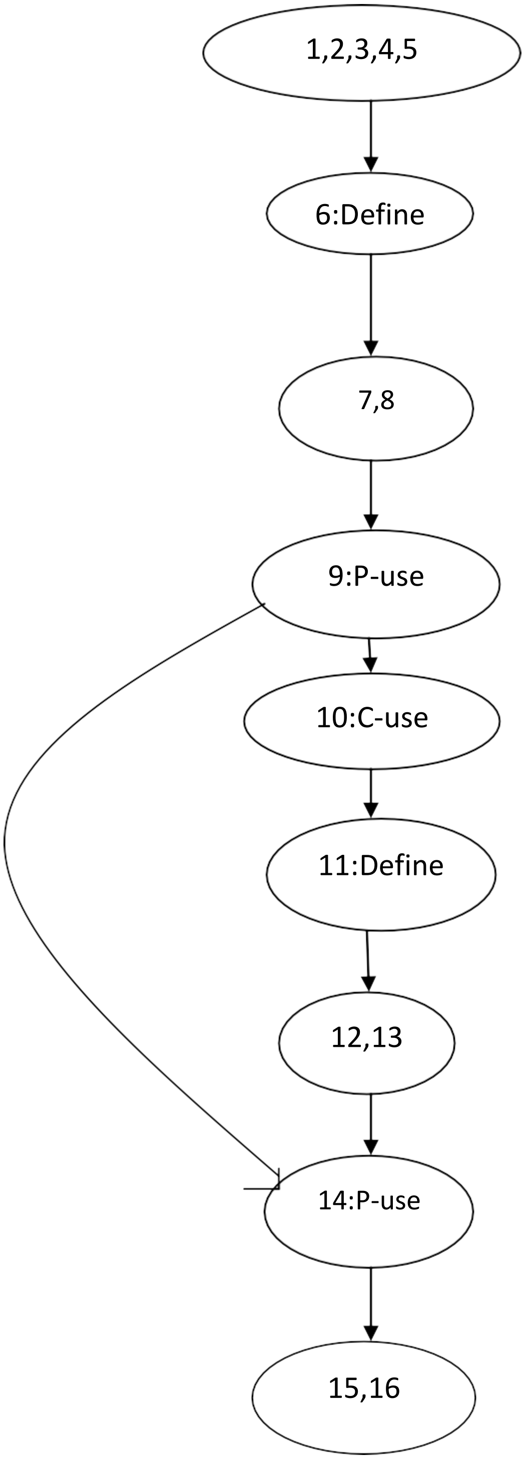
DC-PATHS:-

- A path (i, n_1, ..., n_m, j) is called a *definition-clear path* with respect to x from node i to node j if it contains no definitions of variable x in nodes (n_1, ..., n_m , j) .
- The family of data flow criteria requires that the test data execute definition-clear paths from each node containing a definition of a variable to specified nodes containing *c-use* and edges containing *p-use* of that variable**.**

CONTROL FLOW GRAPH:-

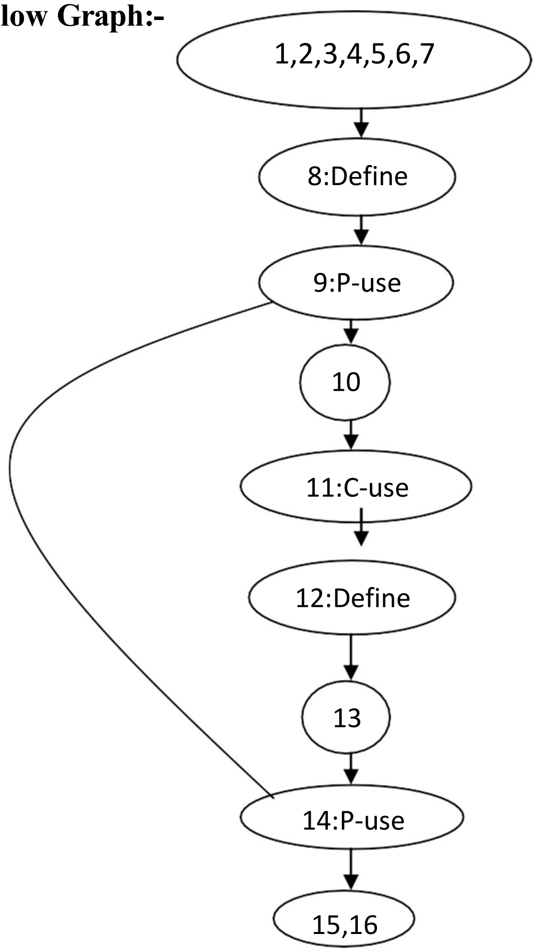


Variable „a‟ Data Flow Graph:-



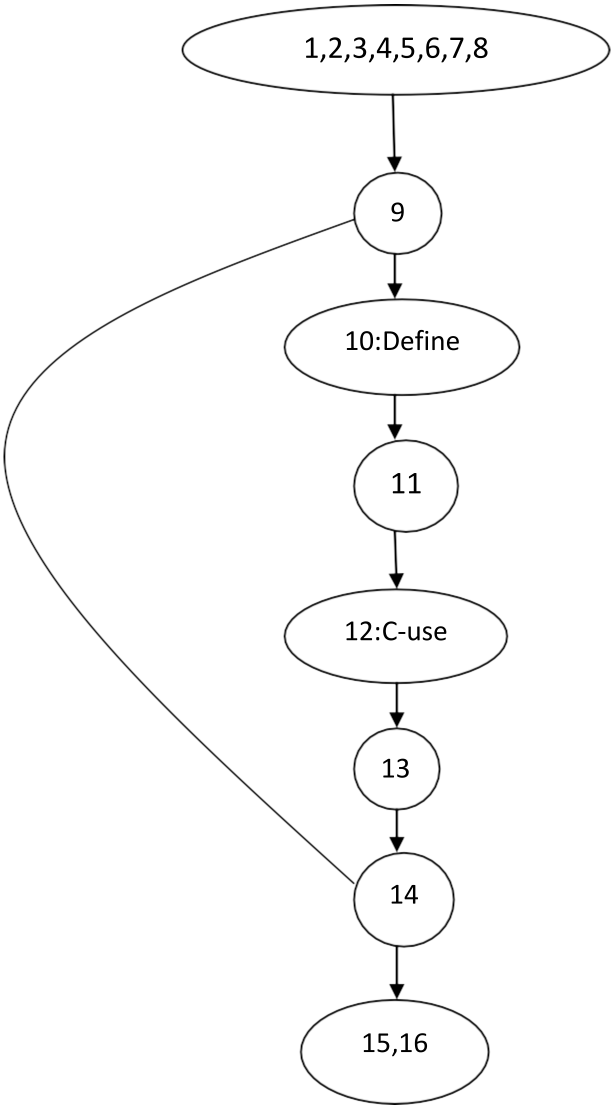
DU-PATHS	DC-PATHS

Variable „b‟ Data Flow Graph:-



DU-PATHS	DC-PATHS

Variable „t‟ Data Flow Graph:-



For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

DU-PATHS	DC-PATHS
1.	1.

Variable	Defined at	Used at
A		
b		
t		

VIVA VOCE QUESTIONS:

1. What is data flow testing?
2. What are the different data flow anomalies?
3. What is the difference definition node & usage node?
4. What is difference between static data flow testing & dynamic data flow testing?
5. What is du-path?
6. What is dc-path?
7. What are the different variants of Inspection process?

EXPERIMENT-11

AIM: Consider the program and generate possible program slices for all variables. Design at least one test case from every slice.

```
#include<stdio.h>
#include<conio.h>
3. void main ()
4. {
3 int a, b, t;
14.     Clrscr ();
15.     Printf (“Enter first number”);
16.     scanf (“%d”,&a);
17.     printf(“Enter second number”);
18.     scanf(“%d”,&b);
19.     if (a<b){
20. t=a;
21. a=b;
22. b=t;
23. }
14. printf (“%d %d”, a, b);
15 getch ();
}
```

DESCRIPTION:

One of the program analysis techniques is program slicing. The main applications of program slicing include various software engineering activities such as program understanding, debugging, testing, program maintenance, complexity measurement and so on. Program slicing is a feasible method to restrict the focus of a task to specific sub-components of a program. It can also be used to extract the statements of a program that are relevant to a given computation.

TEST CASES:-

Test Case id	a	b	a	b
T1	10	20	20	10
T2	20	10	20	10
T3	10	10	10	10

EXECUTION SLICE:-

The Set of statements executed under a Test Case is called as an Execution Slice of a program.

```
#include<stdio.h>
#include<conio.h>
1.void main ()
2.{
3.int a, b, t;
4.Clrscr ();
5.Printf (“Enter first number”);
6.scanf (“%d”,&a);
7.printf(“Enter second number”);
8.scanf(“%d”,&b);
9.if (a<b){
10.t=a;
11.a=b;
12. b=t;
13.}
14. printf (“%d %d”, a, b);
15 getch ();
}
```

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

In the above Program the IF statement(9) is changed to **“if(a==b)”** then only the test case T3 has to be re-run.

Test Case id	a	B	a	b
T3				

DYNAMIC SLICE:-

The set of statements executed under a test case and having an effect on program output is called as Dynamic Slice of a program.

```
#include<stdio.h>
#include<conio.h>
1void main ()
2{
3.int a, b, t;
4.Clrscr ();
5.Printf (“Enter first number”);
6.scanf (“%d”,&a);
7.printf(“Enter second number”);
8.scanf(“%d”,&b);
9.if (a<b){
10.t=a;
11.a=b;
12. b=t;
13.}
14. printf (“%d %d”, a, b);
15 getch ();
}
```

In the above Program the IF statement(9) is changed to **"if(a>b)"** then only the test case T2 has to be re-run.

Test Case id	a	B	a	b
T2				

VIVA VOCE QUESTIONS:

1. What is Regression Testing?
2. Difference between Progressive & Regressive Testing?
3. What are the objectives of regression testing?
4. What are the types of Test Case Prioritization?

5.What are the different Prioritization techniques?

6. What is execution slice?

7. What is dynamic slice?

8. What is relevant slice?

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

EXPERIMENT-12

AIM: Consider the code to arrange the nos. in ascending order. Generate the test cases for relational coverage, loop coverage and path testing. Check the adequacy of the test cases through mutation testing and also compute the mutation score for each.

PROGRAM:

```
1.i = 0;
2.n=4;
3.While (i<n-1)
4.do j = i + 1;
5.While (j<n)
6.do if A[i]<A[j]
7. Swap (A[i], A[j]);
8 end do;
9. i=i+1;
10. end do
```

DESCRIPTION:

PATH TESTING:

Path Testing is a structural testing method based on the source code or algorithm and NOT based on the specifications. It can be applied at different levels of granularity.

PATH TESTING TECHNIQUES:

- **Control Flow Graph (CFG) -** The Program is converted into Flow graphs by representing the code into nodes, regions and edges.
- **Decision to Decision path (D-D) -** The CFG can be broken into various Decision to Decision paths and then collapsed into individual nodes.

- **Independent (basis) paths -** Independent path is a path through a DD-path graph which cannot be reproduced from other paths by other methods.

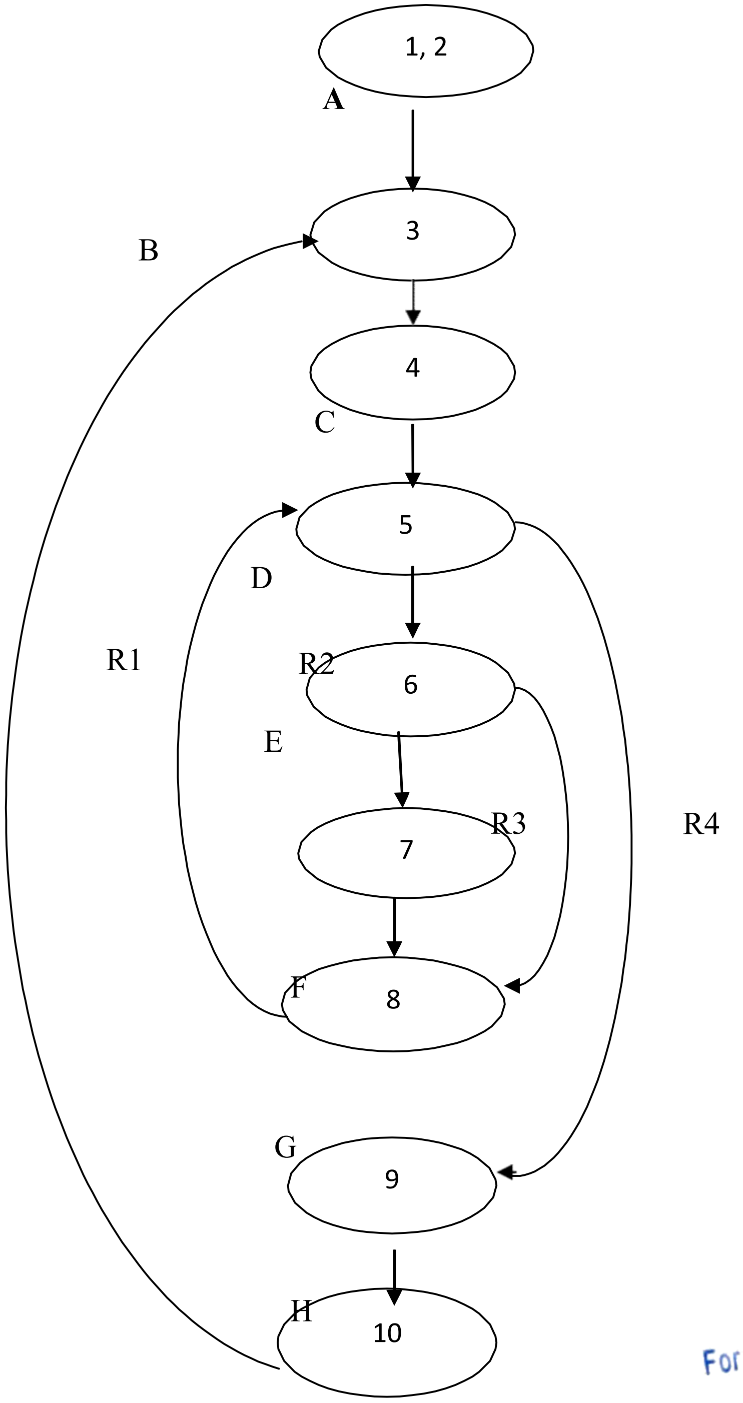
MUTATION TESTING:

Mutation testing is a structural testing technique, which uses the structure of the code to guide the testing process. On a very high level, it is the process of rewriting the source code in small ways in order to remove the redundancies in the source code.

These ambiguities might cause failures in the software if not fixed and can easily pass through testing phase undetected.

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

Control Flow Graph:



For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

Cyclomatic Complexity:

Cyclomatic Complexityis measured in three ways:

1. Cyclomatic complexity = E - N + P

   Where, E = number of edges in the flow graph.

   N = number of nodes in the flow graph.

   P = number of Procedures in the flow graph

   V (G) = E - N + P

   =

2. Cyclomatic complexity = number of regions in graph

   V (G)=

3. Cyclomatic complexity = number of Decision nodes in the graph + 1

   V (G) =

IndependentPaths:

The number of different paths from the starting node of the flow graph to reach the ending node of the flow graph. A graph may have 1 or more independent paths.

TEST CASES:

TID	INPUT	EXPECTED OUTPUT	OUTPUT	INPUTPATH
TC01				
TC02				
TC03				

ORIGINAL PROGRAM:

```
i = 0;
n=4;
While (i<n-1)
do j = i + 1;
    While (j<n)
    do if A[i]<A[j]
        Swap (A[i], A[j]);
    end do;
    i=i+1;
end do
```

MUTATED PROGRAM:

```
i = 0;
n=4;
While (i<n+1) // M1
do j = i + 1;
    While (j<n+1) // M2
```

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

```
do if A[i]>=A[j] // M3
Swap (A[i], A[j]);
end do;
i=i+1;
end do
```

KILLED MUTANT:

If the original program and mutant programs generate the same output, then that mutant is killed by the test case. Hence the test case is good enough to detect the change between the original and the mutant program.

LIVE MUTANT:

If the original program and mutant program generate different output, Mutant is kept alive. In such cases, more effective test cases need to be created that kill all mutants.

TEST CASES:

TID	INPUT	ORIGINAL OUTPUT	MUTANT OUPUT	MUTANT
TC01				
TC02				
TC03				

MUTATION SCORE:

The mutation score is defined as the percentage of killed mutants with the total number of mutants.

Mutation Score=(Killed mutants/total number of mutants)*100

=

From the above we can conclude that Mutant score is_______,so that the above test cases are ________test cases.

VIVA VOCE QUESTIONS:

1. What is Path Testing?
2. What is Loop Testing?
3. What are the test cases to be considered while loop testing?
4. What is Mutant?
5. Difference between Primary Mutants & Secondary Mutants?
6. What is meant by live mutant?
7. What is meant by killed mutant?
8. How do you calculate the Mutation Score?

EXPERIMENT-13

AIM: Design and develop a program in a language of your choice to solve the triangle problem defined as follows : Accept three integers which are supposed to be the three sides of triangle and determine if the three values represent an equilateral triangle, isosceles triangle, scalene triangle, or they do not form a triangle at all. Derive test cases for your program based on decision-table approach, execute the test cases and discuss the results

PROGRAM:

```
#include<stdio.h>
int main()
{
int a,b,c;
char istriangle;
printf("enter 3 integers which are sides of triangle\n");
scanf("%d%d%d",&a,&b,&c);
printf("a=%d\t,b=%d\t,c=%d",a,b,c);
if( a<b+c && b<a+c && c<a+b )     // to check is it a triangle or not
istriangle='y';
else
istriangle ='n';
if (istriangle=='y')
if ((a==b) && (b==c))
printf("equilateral triangle\n");
else if ((a!=b) && (a!=c) && (b!=c))
printf("scalene triangle\n");
else
printf("isosceles triangle\n");
else
printf("Not a triangle\n");
return 0;
}
```

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

DESCRIPTION:

DECISION TABLE BASED TESTING:

A decision table is an excellent tool to use in both testing and requirements management. Essentially it is **a structured exercise to formulate requirements when dealing with complex business rules**. Decision tables are used to model complicated logic.

In a decision table, **conditions are usually expressed as true (T) or false (F)**. Each column in the table corresponds to a rule in the business logic that describes the unique combination of circumstances that will result in the actions.

Decision tables can be used in all situations where the outcome depends on the combinations of different choices, and that is usually very often. In many systems there are tons of business rules where decision tables add a lot of value.

Steps for Constructing Decision Table and generating Test cases:

1. **Analyze the requirement and create the first column**
2. **Add Columns**
3. **Reduce the Table**
4. **Determine Actions**
5. **Write Test Cases**

DECESION TABLE:

RULES		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
Condition Stub	C1:a<b+c	F	T	T	T	T	T	T	T	T	T	T
	C2: b<a+c	-	F	T	T	T	T	T	T	T	T	T
	C3: c<a+b	-	-	F	T	T	T	T	T	T	T	T
	C4: a=b	-	-	-	T	T	T	T	F	F	F	F
	C5: a=c	-	-	-	T	T	F	F	T	T	F	F
	C6:b=c	-	-	-	T	F	T	F	T	F	T	F
Action Stub	A1: Not a Triangle	X	X	X								
	A2: Scalene Triangle											X
	A3: Isosceles Triangle							X		X	X	
	A4: Equilateral Triangle				X							
	A5: Impossible					X	X		X			

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

TEST CASES:

Test case ID	Description	Input			Expected Output	Actual Output	Status
		a	b	C			
TC01							
TC02							
TC03							
TC04							
TC05							
TC06							

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

EXPERIMENT-14

AIM: Understand The Automation Testing Approach (Theory Concept)

DESCRIPTION:

Automation Automation is making a process automatic eliminating the need for human intervention. It is a self-controlling or self-moving process. Automation Software offers automation wizards and commands of its own in addition to providing a task recording and re-play capabilities. Using these programs you can record an IT or business task.

Benefits of Automation

- Fast
- Reliable
- Repeatable
- Programmable
- Reusable
- Makes Regression testing easy
- Enables 24*78 Testing Robust verification.

INTRODUCTION TO SELENIUM

1. History of Selenium

- In 2004 invented by Jason R. Huggins and team.
- Original name is JavaScript Functional Tester [JSFT]
- Open source browser based integration test framework built originally by Thoughtworks.
- 100% JavaScript and HTML
- Web testing tool
- That supports testing Web 2.0 applications

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

- Supports for Cross-Browser Testing(ON Multiple Browsers)
- And multiple Operating Systems
- Cross browser – IE 6/7, Firefox .8+, Opera, Safari 2.0+

2. What is Selenium?

- Acceptance Testing tool for web-apps
- Tests run directly in browser
- Selenium can be deployed on Windows, Linux, and Macintosh.
- Implemented entirely using browser technologies -

□ JavaScript

□ DHTML

□ Frames

3. Selenium Components

- Selenium IDE
- Selenium Core
- Selenium RC

3.1 Selenium IDE

- The Selenium-IDE (Integrated Development Environment) is the tool you
- use to develop your Selenium test cases.
- It is Firefox plug-in
- Firefox extension which allows record/play testing paradigm
- Automates commands, but asserts must be entered by hand
- Creates the simplest possible Locator
- Based on Selenese

3.1.1 OVERVIEW OF SELENIUM IDE:

A. Test Case Pane:

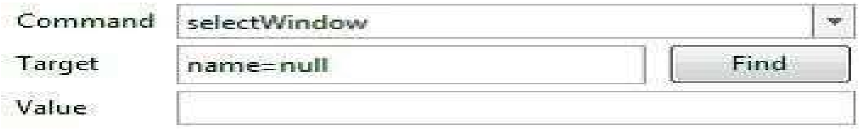
· Your script is displayed in the test case pane.

· It has two tabs.

· one for displaying the command (source)

· and their parameters in a readable “table” format.

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)



Command: selectWindow

Target: name=null

Value:

B. Toolbar: The toolbar contains buttons for controlling the execution of your test cases, including a step feature for

C. Menu Bar:

- **File Menu:** The File menu allows you to create, open and save test case and test suite files.
- **Edit Menu:** The Edit menu allows copy, paste, delete, undo and select all operations for editing the commands in your test case.
- **Options Menu:** The Options menu allows the changing of settings. You can set the timeout value for certain commands, add user-defined user extensions to the base set of Selenium commands, and specify the format (language) used when saving your test cases.

D. Help Menu:

INTRODUCING SELENIUM COMMANDS

The command set is often called selenese. Selenium commands come in three “flavors”:

Actions, Accessory and **Assertions.**

a. **Actions:** user actions on application / Command the browser to do something.

Actions are commands that generally manipulate the state of the application.

1. Click link- click / Clickandwait
2. Selecting items

b. **Accessors**: Accessors examine the state of the application and store the results in variables, e.g. "storeTitle".

c. **Assertions:** For validating the application we are using Assertions

1. For verifying the web pages
2. For verifying the text
3. For verifying alerts

Assertions can be used in 3 modes:

For Techno India NJR Institute of Technology

पंकज पोरवाल

Dr. Pankaj Kumar Porwal

(Principal)

- assert
- verify
- waitFor

Example: "assertText","verifyText" and "waitForText".

NOTE:

1. When an "assert" fails, the test is aborted.
2. When a "verify" fails, the test will continue execution
3. "waitFor" commands wait for some condition to become true

COMMONLY USED SELENIUM COMMANDS

These are probably the most commonly used commands for building test.

open - opens a page using a URL.

click/clickAndWait - performs a click operation, and optionally waits for a new page to load.

verifyTitle/assertTitle - verifies an expected page title.

verifyTextPresent- verifies expected text is somewhere on the page.

verifyElementPresent -verifies an expected UI element, as defined by its HTML tag, is present on the page.

verifyText - verifies expected text and it‟s corresponding HTML tag are present on the page.

verifyTable - verifies a table‟s expected contents.

waitForPageToLoad -pauses execution until an expected new page loads. Called automatically when clickAndWait is used.

waitForElementPresent -pauses execution until an expected UI element, as defined by its HTML tag, is present on the page.

3.1.2 RECORDING AND RUN SETTINGS

When Selenium-IDE is first opened, the record button is ON by default.

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

During recording, Selenium-IDE will automatically insert commands into your test case based on your actions.

a. **Remember Base URL MODE** - Using Base URL to Run Test Cases in Different Domains

b. **Record Absolute recording mode** – Run Test Cases in Particular Domain.

3.1.3 RUNNING TEST CASES

Run a Test Case Click the Run button to run the currently displayed test case. **Run a Test Suite** Click the Run All button to run all the test cases in the currently loaded test suite.

Stop and Start The Pause button can be used to stop the test case while it is running. The icon of this button then changes to indicate the Resume button. To continue click Resume.

Stop in the Middle You can set a breakpoint in the test case to cause it to stop on a particular command. This is useful for debugging your test case. To set a breakpoint, select a command, right-click, and from the context menu select Toggle Breakpoint.

Start from the Middle You can tell the IDE to begin running from a specific command in the middle of the test case. This also is used for debugging. To set a startpoint, select a command, right-click, and from the context menu select Set/Clear Start Point.

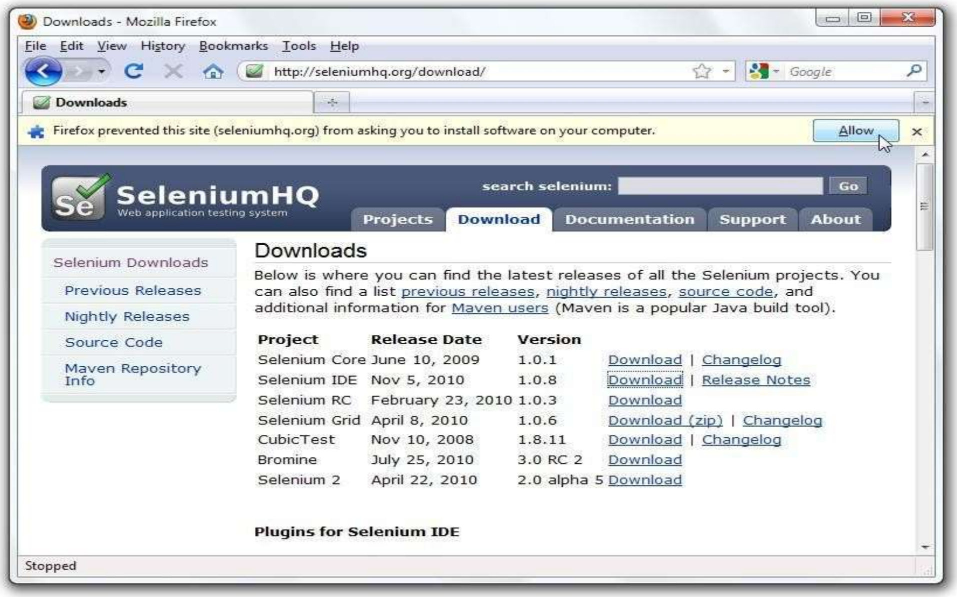
Run Any Single Command Double-click any single command to run it by itself. This is useful when writing a single command. It lets you immediately test a command you are constructing, when you are not sure if it is correct. You can double-click it to see if it runs correctly. This is also available from the context menu.

Test Suite:

A test suite is a collection of tests. Often one will run all the tests in a test suite as one continuous batch-job. When using Selenium-IDE, test suites also can be defined using a simple HTML file. The syntax again is simple. An HTML table defines a list of tests where each row defines the filesystem path to each test.

INSTALLING THE IDE

Using Firefox, first, download the IDE from the SeleniumHQ downloads page Firefox will protect you from installing addons from unfamiliar locations, so you will need to click „Allow" to proceed with the installation, as shown in the following screenshot.



Firefox prevented this site (seleniumhq.org) from asking you to install software on your computer.

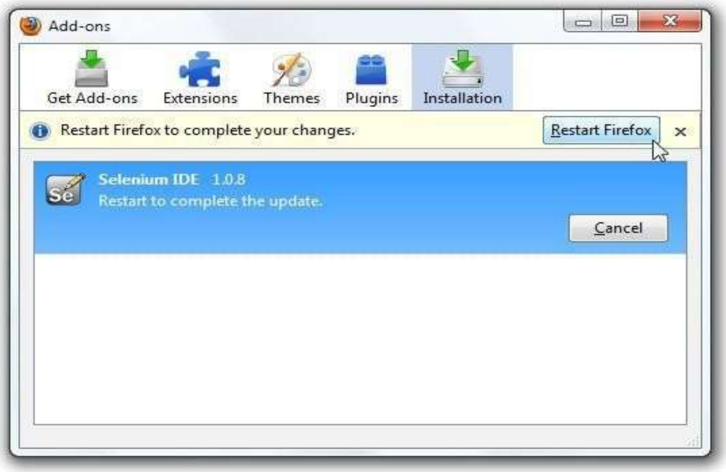
Downloads

Below is where you can find the latest releases of all the Selenium projects. You can also find a list previous releases, nightly releases, source code, and additional information for Maven users (Maven is a popular Java build tool).

Project	Release Date	Version	
Selenium Core	June 10, 2009	1.0.1	Download | Changelog
Selenium IDE	Nov 5, 2010	1.0.8	Download | Release Notes
Selenium RC	February 23, 2010	1.0.3	Download
Selenium Grid	April 8, 2010	1.0.6	Download (zip) | Changelog
CubicTest	Nov 10, 2008	1.8.11	Download | Changelog
Bromine	July 25, 2010	3.0 RC 2	Download
Selenium 2	April 22, 2010	2.0 alpha 5	Download

Plugins for Selenium IDE

Select Install Now. The Firefox Add-ons window pops up, first showing a progress bar, and when the download is complete, displays the following.



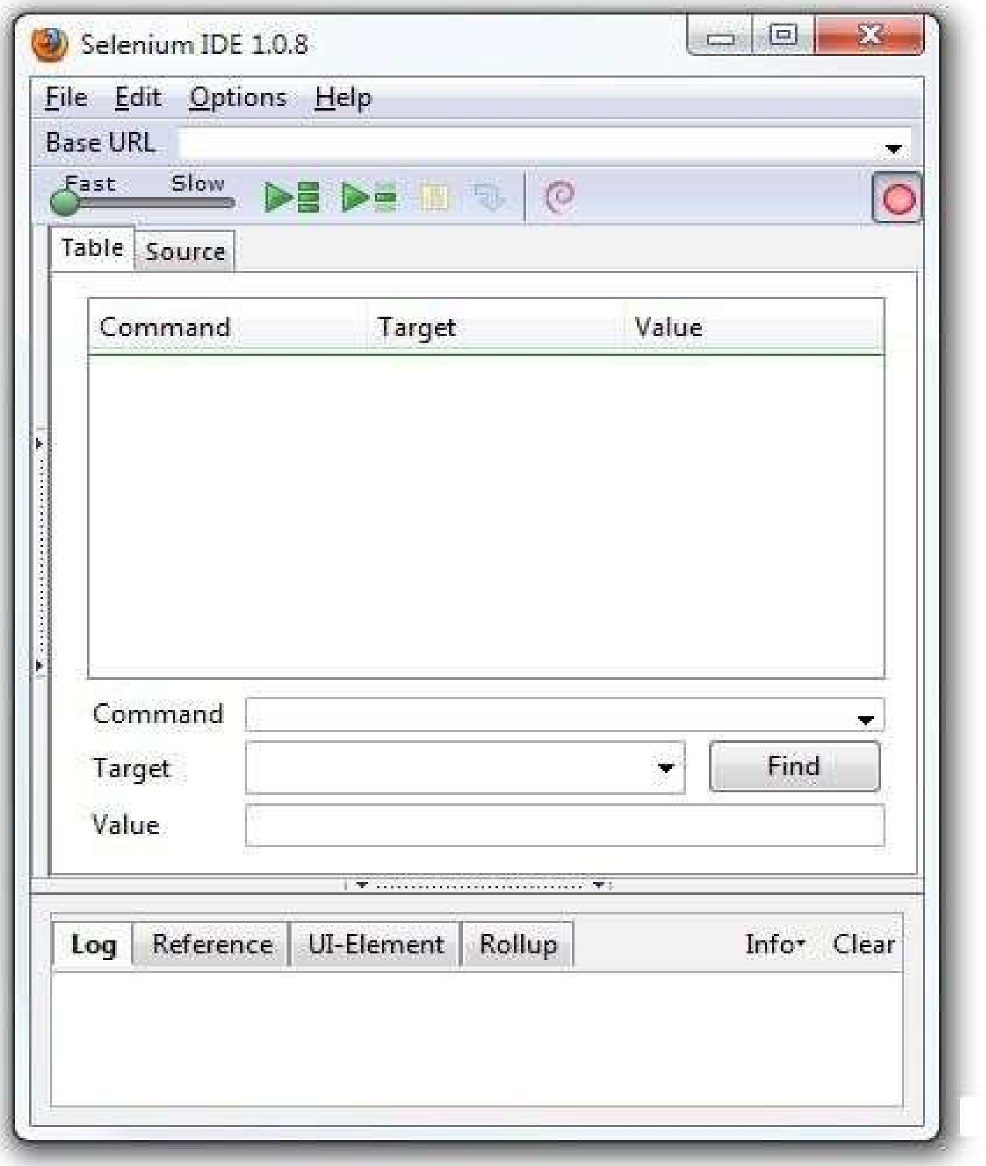
Restart Firefox to complete your changes.

Selenium IDE 1.0.8
Restart to complete the update.

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

Restart Firefox. After Firefox reboots you will find the Selenium-IDE listed under the Firefox Tools menu.

When downloading from Firefox, you‟ll be presented with the following window.



For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)