Parul University

FACULTY OF ENGINEERING AND TECHNOLOGY
BACHELOR OF TECHNOLOGY

Software Testing and Quality Assurance Laboratory

(203105396)

7TH SEMESTER

Computer Science & Engineering Department

Laboratory Manual

FACULTY OF ENGINEERING AND TECHNOLOGY
Subject Name: Software Testing and Quality Assurance
Subject Code: 203105396
B.Tech. 4TH year 7TH semester

CERTIFICATE

This is to certify that Mr./Ms ..... with enrollment no……has successfully completed his/her laboratory experiments in the subject (subject code) of …….. from the department of ……. in …… during the academic year ……

Date of Submission: ..... Staff In Charge: ……..

Head of Department: .....

TABLE OF CONTENT

PRACTICAL – 1

AIM : Create test cases using boundary value analysis.

Boundary value analysis:

Boundary Value Analysis is based on testing the boundary values of valid and invalidpartitions. The behaviour at the edge of the equivalence partition is more likely to be incorrect than the behaviour within the partition, so boundaries are an area where testing is likely to yield defects.

It checks for the input values near the boundary that have a higher chance of error. Every partition has its maximum and minimum values and these maximum and minimum values arethe boundary values of a partition.

- **Date format check**

  If Date should be in duration of 10 days i.e 20-11-22 to 30-11-22

Invalid	Valid	Invalid
19-11-22	20-11-22,21-11-22,29-11-22,30-11-22	1-12-22

Test Case 1 = 19-11-22 (invalid)

Test Case 2 = 20-11-22 (valid)

Test Case 3 = 21-11-22 (valid)

Test Case 4= 29-11-22 (valid)

Test Case 5 = 30-11-22 (valid)

Test Case 6 = 1-12-22 (invalid)

Test Scenario ID	Date check	Test Case ID	Date check-1A
Test Case Description	-	Test Priority	High
Pre-Requisite	NA	Post-requisite	NA

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Result
1	Date check	21-11-22	Date valid	Date valid	Pass
2	Date check	19-11-22	Date invalid	Date invalid	Fail

➢ Weight limit range

If there is health survey form for particular set of people and there is a input columnfor weight and condition is there for a valid range (e.g 40 kg - 85kg).

Invalid	Valid	Invalid
39	40,41,84,85	86

Test Case 1 = 39 (invalid)

Test Case 2 = 40 (valid)

Test Case 3 =41 (valid)

Test Case 4= 84 (valid)

Test Case 5 = 85 (valid)

Test Case 6 = 86 (invalid)

Test Scenario ID	Weight check	Test Case ID	Weight check-1
Test Case Description		Test Priority	High
Pre-Requisite	NA	Post-requisite	NA

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Result
1	Weigh check	41kg	Valid	Valid	Pass
2	Weight check	102kg	Invalid	Invalid	Fail

➢ Media file size range

If only media files of size between 50kb and 1mb should be uploaded.

Invalid	Valid	Invalid
49kb	50kb, 51kb, 0.99mb,1mb	1.1mb

Test Case 1 = 49kb (invalid)
Test Case 2 = 50kb (valid)
Test Case 3 = 51kb (valid)
Test Case 4= 0.99mb (valid)
Test Case 5 = 1mb (valid)
Test Case 6 = 1.1mb (invalid)

Test Scenario ID	Size check	Test Case ID	Size check-1
Test Case Description		Test Priority	High
Pre-Requisite	NA	Post-requisite	NA

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Result
1	File Size check	51kb	Valid	Valid	Pass
2	File Size check	1.2mb	Invalid	Invalid	Fail

PRACTICAL - 2

AIM : Create test cases using equivalence partitioning.

Equivalence Class Partitioning:

Equivalence Partitioning Method is also known as Equivalence class partitioning (ECP). It isa software testing technique or black-box testing that divides input domain into classes of data, and with the help of these classes of data, test cases can be derived. An ideal test case identifies class of error that might require many arbitrary test cases to be executed before general error is observed.

In equivalence partitioning, equivalence classes are evaluated for given input conditions.Whenever any input is given, then type of input condition is checked, then for this input conditions, Equivalence class represents or describes set of valid or invalid states.

Guidelines for Equivalence Partitioning:

- ✓ If the range condition is given as an input, then one valid and two invalid equivalenceclasses are defined.
- ✓ If a specific value is given as input, then one valid and two invalid equivalence classes aredefined.
- ✓ If a member of set is given as an input, then one valid and one invalid equivalence class isdefined.
- ✓ If Boolean no. is given as an input condition, then one valid and one invalid equivalenceclass is defined.

- ➢ **CVV value in payment gate away section**

Invalid	Valid	Invalid
12	123	1234

Valid Input : 3 digits

Invalid Input : 1 digit, 2 digits

Valid Class : Enter 3 digit cvv = 123

Invalid Class : Enter cvv which has more than 3 digits or less than 3 = 12, 1234, 1

- ➢ **BMI value range**

If input section is asking for BMI value and it needs a range (for e.g 50-60)

Invalid	Valid	Invalid
44	56	63

Valid Input : Any Input between range 50-60

Invalid Input : Any Input between range <=49 or >=61

Valid Class : Enter BMI = 56

Invalid Class : Enter BMI which has more than 60 digits or less than 50 = 44, 46, 64, 63

- **Boolean Value**

Have you already created account

Invalid	Valid
No	Yes

v

alid Input : Yes
Invalid Input: no
Valid Class : Have you created account = Yes
Invalid Class : Have you created account = No

PRACTICAL – 3

AIM : Design independent paths by calculating cyclomatic complexityusing date problem.

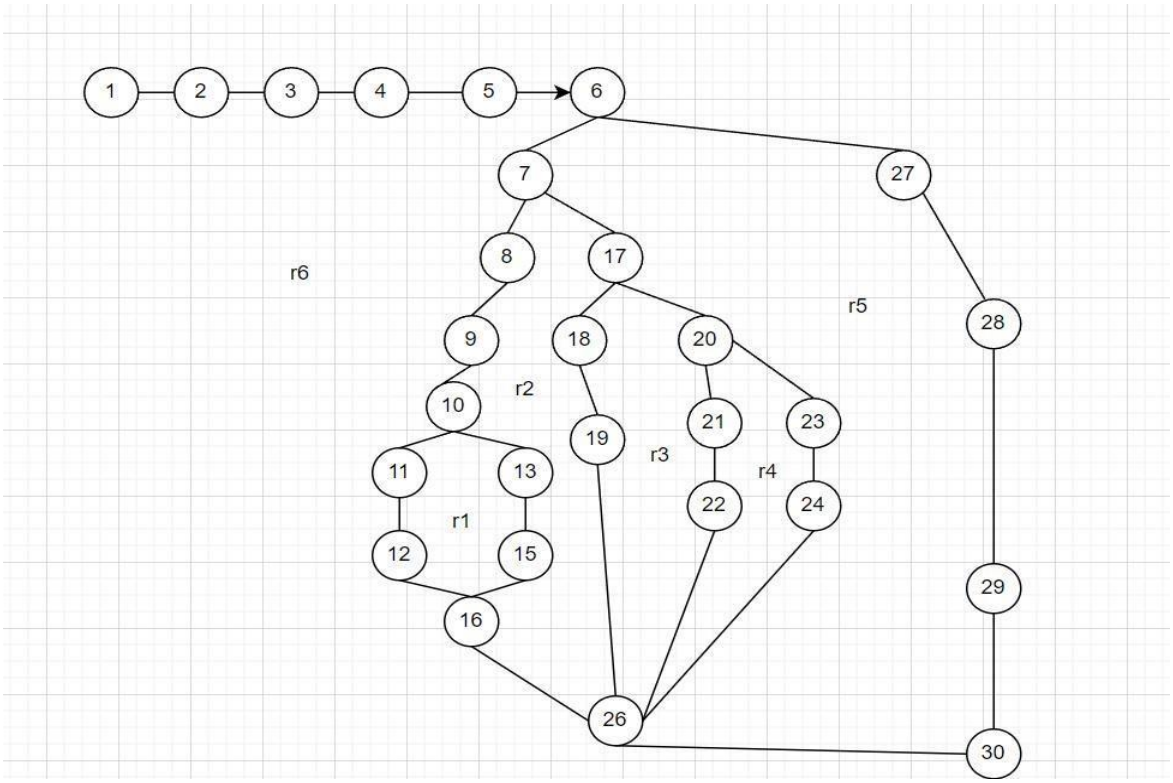
Cyclomatic complexity: It is software metric used to measure the complexity of a program. Thomas J. McCabe developed this metric in 1976.McCabe interprets a computer program as a set of a strongly connected directed graph. Nodes represent parts of the source code having no branches and arcs represent possible control flow transfers during program execution. Thenotion of program graph has been used for this measure, and it is used to measure and controlthe number of paths through a program. The complexity of a computer program can be correlated with the topological complexity of a graph.

Date problem Code :

```
1. #include <iostream>
2. using namespace std;
3. int main(){
4. int month;
5. cin>>month;
6. if( month>=1 && month<=12 ){
7. if(month==2){
8. int year;
9. cin>>year;
10. if(year%4==0 || year%400==0){
11. cout<<"no of days = 29"<<endl;
12. }
13. else{
14. cout<<"no of days = 28"<<endl;
15. }
16. }
17. else if(month <=7 && month%2!=0){
18. cout<<"no of days = 31"<<endl;
19. }
20. else if(month >7 && month%2==0){
21. cout<<"no of days = 31"<<endl;
22. }
23. else{
```

```
24. cout<<"no of days = 30"<<endl;
25. }
26. }
27. else{
28. cout<<"invalid input"<<endl;
29. }
30. }
```

Control Flow Graph :



Cyclomatic complexity of code using control flow graph is :

Method 1 : Edge - nodes + 2 => 34 - 30 + 2 = 6

Method 2 : Predicate nodes +1 = 5+1 = 6 (6,7,10,17,20 are predicate nodes)

Method 3 : Total regions = 6 (Here r1, r2, r3, r4, r5 & r6 are the all regions)

V(G) = 6 and is same by all the three methods

Identification of Independent paths:

- 1-6-7-8-9-10-11-12-16-26-30
- 1-6-7-8-9-10-13-15-16-26-30
- 1-6-7-17-18-19-26-30
- 1-6-7-17-20-21-22-26-30
- 1-6-7-17-20-23-24-26-30
- 1-6-27-28-29-30

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Result
1	No. of days check	Month: 1	31	31	Pass
2	No. of days check	Month: 8	31	31	Pass
3	No. of days check	Month: 2 Year: 2003	28	28	Pass
4	No. of days check	Month: 3	31	31	Pass
5	No. of days check	Month: 2 Year: 2004	29	29	Pass

PRACTICAL -4

AIM: Design test cases using Decision table

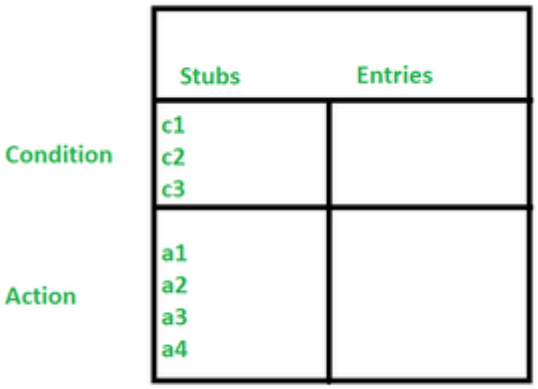
Theory :

What is a Decision Table :

Decision tables are used in various engineering fields to represent complex logical relationships. This testing is a very effective tool in testing the software and its requirements management. The output may be dependent on many input conditions and decision tables give a tabular view of various combinations of input conditions and these conditions are in the form of True(T) and False(F). Also, it provides a set of conditions and its corresponding actions required in the testing.

Parts of Decision Tables :

In software testing, the decision table has 4 parts which are divided into portions and are given below :



1. **Condition Stubs :** The conditions are listed in this first upper left part of the decision table that is used to determine a particular action or set of actions.
2. **Action Stubs :** All the possible actions are given in the first lower left portion (i.e, below condition stub) of the decision table.
3. **Condition Entries :** In the condition entry, the values are inputted in the upper right portion of the decision table. In the condition entries part of the table, there are multiple rows and columns which are known as Rule.
4. **Action Entries :** In the action entry, every entry has some associated action or set of actions in the lower right portion of the decision table and these values are called outputs.

Types of Decision Tables :

The decision tables are categorized into two types and these are given below:

1. **Limited Entry :** In the limited entry decision tables, the condition entries are restricted to binary values.
2. **Extended Entry :** In the extended entry decision table, the condition entries have more than two values. The decision tables use multiple conditions where a condition may have many possibilities instead of only ‘true’ and ‘false’ are known as extended entry decision tables.

Applicability of Decision Tables :

- The order of rule evaluation has no effect on the resulting action.
- The decision tables can be applied easily at the unit level only.
- Once a rule is satisfied and the action selected, n another rule needs to be examined.
- The restrictions do not eliminate many applications.

Example of Decision Table Based testing :

Below is the decision table of the program for determining the largest amongst three numbers in which its input is a triple of positive integers (x,y, and z) and values are from the interval [1, 300].

Table 1 : Decision Table of largest amongst three numbers :

Conditions	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R 13	R14
c1: x > = 1?	F	T	T	T	T	T	T	T	T	T	T	T	T	T
c2: x <= 300?		F	T	T	T	T	T	T	T	T	T	T	T	T
c3: y > = 1?			F	T	T	T	T	T	T	T	T	T	T	T
c4: x <= 300?				F	T	T	T	T	T	T	T	T	T	T
c5: z > = 1?					F	T	T	T	T	T	T	T	T	T
c6: z <= 300?						F	T	T	T	T	T	T	T	T
c7: x>y?							T	T	T	T	F	F	F	F
c8: y>z?							T	T	F	F	T	T	F	F
c9: z>x?							T	F	T	F	T	F	T	F
Rule Count	256	128	64	32	16	8	1	1	1	1	1	1	1	1
a1 : Invalid input	X	X	X	X	X	X								

a2 : x is largest								X		X			
a3 : y is largest											X	X	
a4 : z is largest									X				X
a5 : Impossible							X						

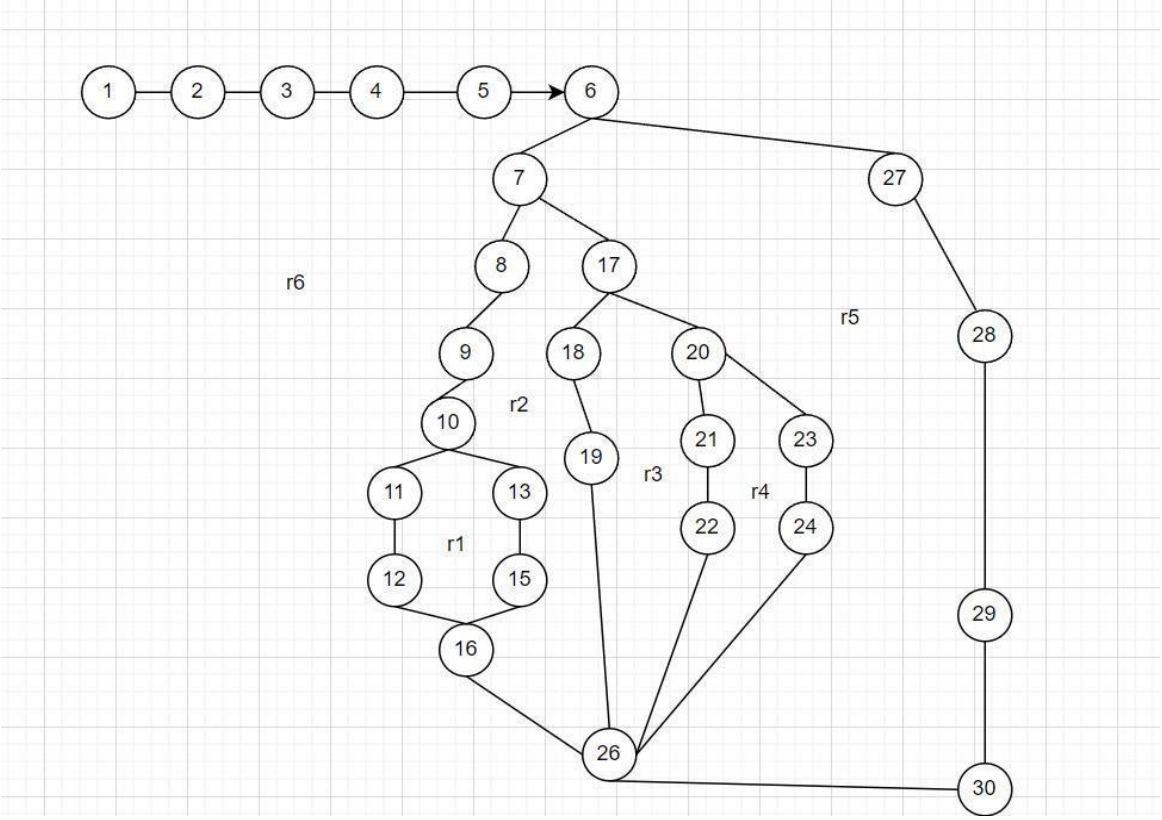
Practical -5

AIM : Design independent paths bytaking DD path using date problem

Theory:

- When we have a flow graph, we can easily draw another graph that is known as decision-to-decision or (DD) path graph, wherein we lay our main focus on the decision nodes only. The nodes of the flow graph arecombined into a single node if they are in sequence.
- DD-path testing is also called C2 testing or branch coverage.

Control flow graph:

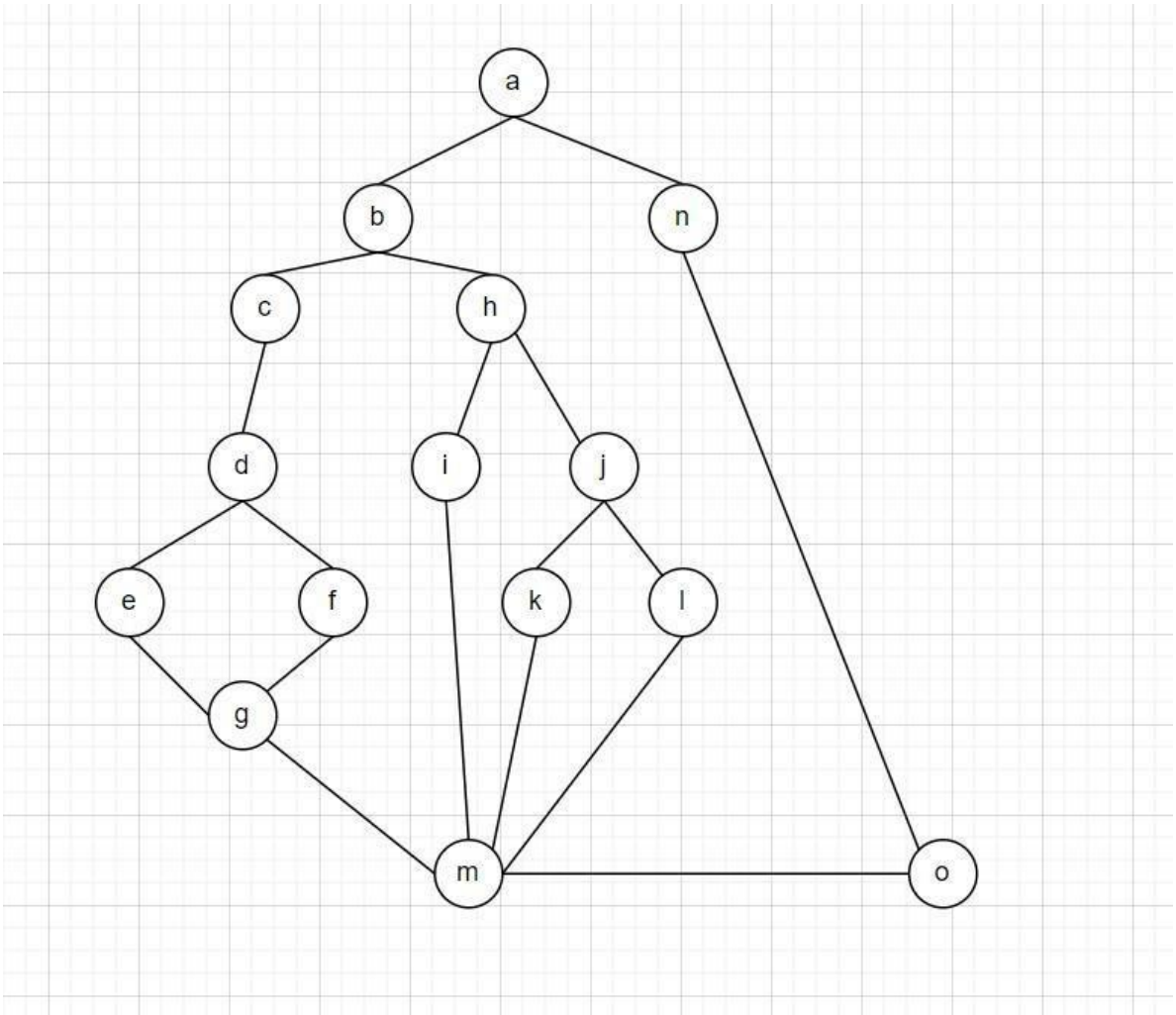


Control flow graph reduction :

1-6	a
7	b
8-9	c
10	d
11-12	e
13-15	f
16	g

17	h
18-19	i
20	j
21-22	k
23-25	l
26	m
27-29	n
30	o

DD path graph :



Identification of Independent path:

- a-b-c-d-e-g-m-o
- a-b-c-d-f-g-m-o
- a-b-h-i-m-o
- a-b-h-j-k-m-o
- a-b-h-j-l-m-o
- a-n-o