Ajay Kumar Garg Engineering College, Ghaziabad

Department of CSE

Model Solution- ODD Semester (2017-18)

Sessional Test -2

Subject Code	:	NCS-071
Subject Name	:	Software Testing and Audit
Names of Faculty Teaching with Signature		1. Ms Deepti Singh [illegible] 2. Ms. Shiva Tyagi 3. Ms. Nistha Ahuja

Name and Signature of Hod: Mamta Bhusry

Prof. Mamta Bhusry

[illegible]

SECTION-A

Q1. what is the significance of Cyclomatic Complexity?

Ans :- Cyclomatic Complexity is a software metric, used to indicate the complexity of a program. It is used to find the linearly independent paths in a source code.

Q2. what is defect seeding?

Ans :- Defect seeding is a technique that was developed to estimate the number of defects resident in a piece of software. In this the defects are injected in the code.

Q3. What is the role of risk matrix for the reduction of test cases?

Ans:- Risk matrix is used to capture identified problems, estimate their probability of occurence with impact and rank the risks based on this information. Risk matrix is divided into quadrants and each quadrants represent priority category.

Q4. Difference between functional and structural testing?

Ans :-

Functional	Structural
(1) It is a black-box testing.	(1) It is a white - box testing.
(2) It basically concern about final result.	(2) It concerns about the result and the process also.

(2) It require execution.	(3) It doesn't requires execution.
eg. are (4) Boundary value Analysis, decision table etc.	(4) eg. are Mutation testing, DD Path graph etc.

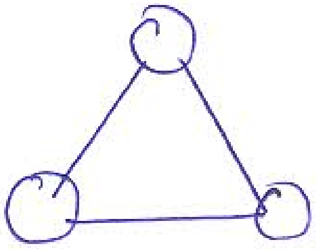
Q5. What are differences between directed graph and undirected graph? Which one is more relevant in software testing and why?

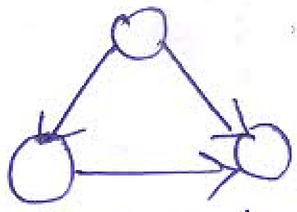
Ans :- A graph consist of set of edges and nodes.

A directed graph is a graph that consist of directions.

An undirected graph is a graph that doesn't consist of any directions.

Directed graph is more relevant in software testing as with the help of directions, we can see the data flow.


undirected


directed

SECTION - B

Q6. Explain the Code coverage prioritization technique. What are the test cases selection criterion? Write the modification algorithm which is used to minimize and prioritize test cases. &

Ans :- A Code coverage technique is based on specific test case prioritization and selects T' from T which is a subset of T. The technique prioritizes test cases of T' and recommends use of high priority test case first and then low priority test cases in descending order.

Test case Selection Criteria :-

This technique identifies those test cases that :-

(i) Execute the modified lines of source code atleast once.

(ii) Execute lines of source code after deletion of deleted lines from the execution history of the test cases and are not-redundant.

Modification Algorithm

The modification Algorithm is used to minimize and prioritize test cases based on the modified lines of source code.

Different Variables used by modification Algorithm are

① T_1	2-D array used to store line numbers.
② modloc	used to store total number of modified lines of source code.
③ mod_locode	1D array to store line numbers of modified lines of source code.
④ nfound	1D array used to store no. of lines of source code matched with modified lines.
⑤ Pos	1D array used to set position of each test case
⑥ Candidate	1D array. Sets to bit 1 corresponding to the position of test case to be removed.
⑦ Priority	1D array used to set the priority of selected test cases.

Q7. How does Regression Testing helps in producing quality software?

Ans:- Regression testing is a procen of re-testing the modified part of the software and ensuring that no new errors have been introduced into previously tested source code due to these modifications. Therefore, regression testing tests both the modified source code and other parts of the source code that may be affected by change.

It helps in producing quality software by:-

(i) Increasing Confidence in the correctness of the modified program.

(ii) Locates errors in modified program.

(iii) preserve the quality and reliability of the software.

(iv) Ensure the software's Continued operation.

Q8. Explain Equivalence class testing Technique. How it is different from Boundary value analysis technique.

Ans :- In Equivalence class testing entire input-domain can be divided into atleast two equivalence classes : One containing all valid inputs and other containing all invalid inputs.

Each equivalence class can further be sub-divided into equivalence classes, on which program is ~~to~~ required to behave differently.

It is different from Boundary Value analysis as in BVA, we select values on or close to boundaries. In BVA we select all valid cases only. whereas in Equivalence class testing we select both valid as well as invalid cases. Eg. If range of x is from [1 to 100] then 5 values are 1, 2, 50, 99, 100.

Eg. of Equivalence class testing is suppose the range is from 1 to 999 then one valid Equivalence class is [1< item < 999] and two invalid equivalence class are [item < 1] and [item > 999]

Q9. Discuss the significance of Decision Table in Testing. What is the purpose of Rule Count? Explain the Concept with the help of an example.

Ans :- Decision Tables are used in many engineering disciplines to represent complex logical relationship.
An Output may be dependent on many Conditions and decision table give a pictorial view of various Combinations of input Conditions.
Don't care Conditions are represented by '-' sign. It has no effect on output.
Ideally each Column has one rule that leads to a test case. A Column in the entry portion of the table is known as RULE.

Rule Count = $2^{\text{no. of don't care conditions.}}$

The term 'rule count' is used with don't care conditions.

Conditions.	R_1	R_2	R_3
C_1	F	T	T
C_2	-	F	T
C_3	-	-	F
Rule Count	4	2	1
a_1	X	X	
a_2			X
a_3	X		

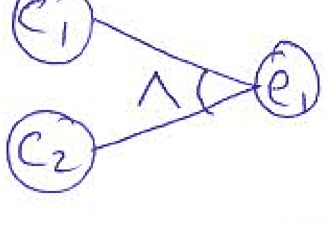
Q10. Write short-Notes on following:-
(i) Mutation Testing,
(ii) Basic Notations and Constraints used in Cause and Effect graph.

Ans :- (i) Mutation Testing :- Mutation testing means to create the mutant version of the Software. Mutant version of the software is the version in which we inject the defects in the software.
Mutation testing is a type of a software testing where we mutate (change) certain statements in the source code and check if the test cases are able to find the errors. It is a type of a White Box-testing.

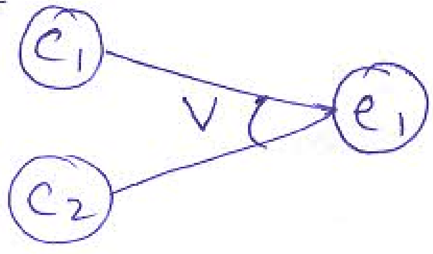
(ii) Basic Notations and constraints in Cause and effect Graph :-

Notations :-

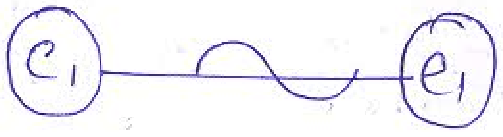
(a) AND :- For effect e_1 to be true, both Causes C_1 and C_2 should be true.



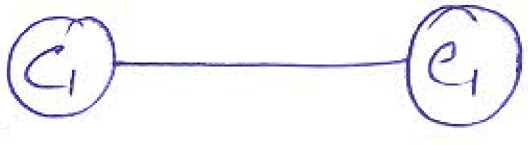
(b) OR :- for effect e_1 to be true, either of causes c_1 or c_2 should be true.



(c) NOT :- for effect e_1 to be true c_1 should be false.



(d) Identity :- If c_1 is 1, then e_1 is 1 else e_1 is 0.



Constraints used in cause and effect Graphs are :-

(1) Exclusion (E) :-
c_1 or c_2 can be 1 [c_1 or c_2 cannot be 1 simultaneously]. However both c_1 or c_2 can be 0, simultaneously.

$$\begin{tikzcd} & \boxed{C_1} \\ E \arrow[ur, dashed, no head] \arrow[dr, dashed, no head] & \\ & \boxed{C_2} \end{tikzcd}$$

(b) <u>Inclusive</u>
Atleast one of C_1 or C_2 must always be 1. Hence both cannot be 0 simultaneously. However, both can be 1.

$$\begin{tikzcd} & \boxed{C_1} \\ I \arrow[ur, dashed, no head] \arrow[dr, dashed, no head] & \\ & \boxed{C_2} \end{tikzcd}$$

(c) <u>One and Only One</u> :-
Only one of C_1 and C_2 must be 1.

$$\begin{tikzcd} & \boxed{C_1} \\ O \arrow[ur, dashed, no head] \arrow[dr, dashed, no head] & \\ & \boxed{C_2} \end{tikzcd}$$

(d) <u>Requires</u> :- for C_1 to be 1, C_2 must be 1.

$$\begin{tikzcd} \boxed{C_1} \arrow[d, dashed, bend right, "R"'] \\ \boxed{C_2} \end{tikzcd}$$

(e) <u>Mask</u> :- e_1 is 1, e_2 is forced to be 0.

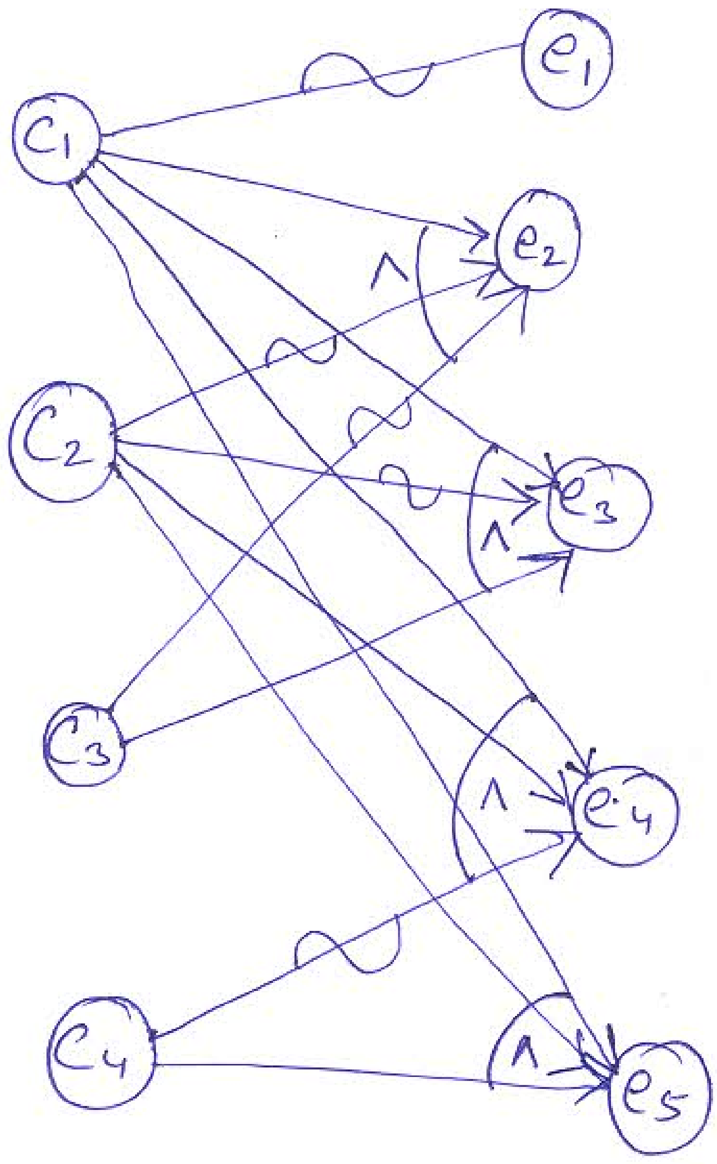
$$\begin{tikzcd} \boxed{e_1} \arrow[d, dashed, bend left, "M"] \\ \boxed{e_2} \end{tikzcd}$$

Section-C

(11) The following conditions must be fulfilled in order to get money from an ATM
→ bankcard is valid
→ PIN must be correctly entered
→ The maximum number of PIN inputs is three
→ There is money in the m/c, and in the account.
The following actions are possible at the m/c :-
→ Reject card
→ Eat card
→ Ask for an alternate dollar amount.
→ Ask for another PIN input.
→ Pay the requested amount of Money.
Draw the Cause-effect graph for the above given problem. Create the decision table and then design test cases.

Ans :- C_1 → Bankcard is Valid
C_2 → PIN is Correct
C_3 → PIN input is 3
C_4 → Money Available

e_1 → Reject Card
e_2 → Ask new pin
e_3 → Eat Card
e_4 → Ask new amount
e_5 → Pay Money



Causes / Conditions Effects/Actions

Decision Table

	R1	R2	R3	R5
C1	0	1	1	X
C2	0	X	X	1
C3	X	0	1	1
C4	X	X	1	1
E1	1	0	0	0
E2	0	1	0	0
E3	0	0	0	0
E4	0	0	1	0
E5	0	0	0	1

0 → False
1 → True

Test - Cases :-

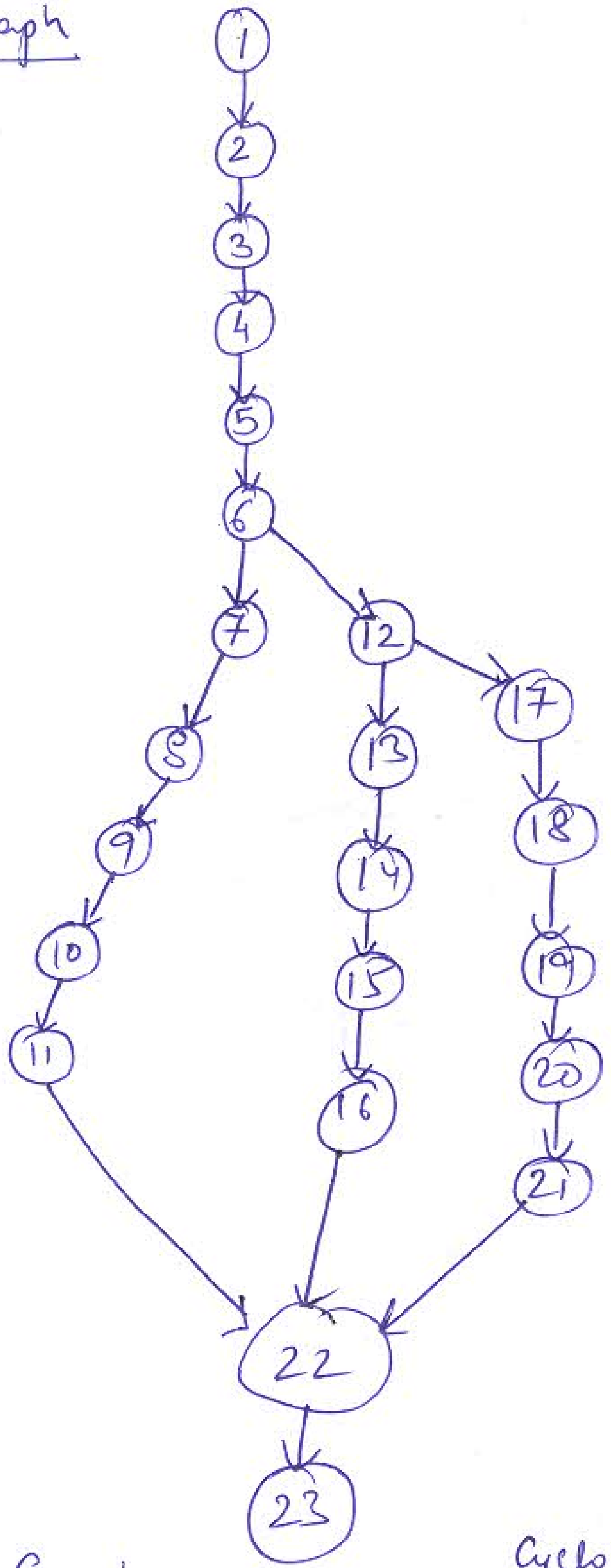
Test Case	Card Valid	Pin Correct -	Pin ~~ip~~ input is 3	Money Available	Expected Output
1.	Yes	Yes	Yes	Yes	Money Paid
2.	Not valid	–	–	–	[Reject Card] Not paid
3.	Yes	No	No	~~y~~ –	Ask new Pin
4.	Yes	Yes	Yes	No.	Ask new amount

Q12. Consider the program to find the roots of quatratic equation.

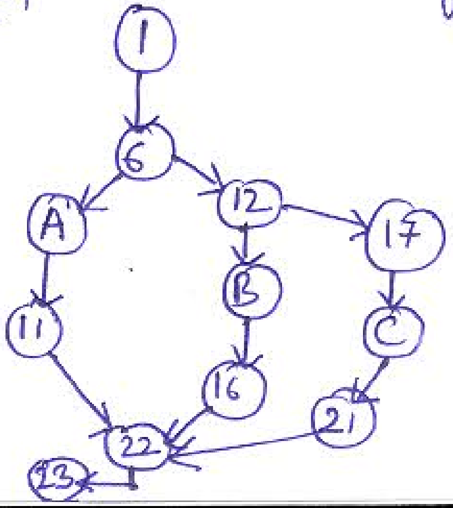
```
    #include <stdio.h>
    #include <math.h>
1.  int main()
2.  { double a, b, c, det, r1, r2, rp, img;
3.  printf("Enter Coffiecient a, b, and c);
4.  scanf("%lf %lf %lf", &a, &b, &c);
5.     det = b*b - 4*a*a;
6.       if (det>0)
7.   { r1= (-b+sqrt(det))/(2*a);
8.
9.     r2= (-b - sqrt(det))/(2*a);
10. printf("r1= %.2lf & r2= %.2lf", r1, r2);
11. }
12.   else if (det==0)
13.   { r1 = r2 = -b/(2*a);
14.
15.    printf("r1 = r2 = %.2lf", r1);
16. }
17.  else
18.       { rp = -b/(2*a);
            img = sqrt(-det)/(2*a);
19.   ~~printf("r1 = r2 = %.2lf", rp, img, rp,~~

20.  printf("r1 = %.2lf + %.2lf and r2= %.2f-
            %.2fi", rp, img, rp, img);
21.   }
22.   return 0;
23.       }
```

a) Draw program graph, DD Path graph. find Cyclomatic complexity. (b) find all du-paths, identify those du path which all not in dc path. Write test cases.

Program Graph



DD Path Graph



Cyclomatic Complexity
= No. of regions
= 3

All du-path in program graph:-

Definition node for all variable is at node 2
use nodes are:-
a → 4, 5, 8, 9, 14, 18, 19
b → 4, 5, 8, 9, 14, 18
c → 4, 5
det → 5, 8, 9, 19
r_1 → 8, 10, 14, 15
r_2 → 10, 14
rp → 20, 18
img → 20, 19

du paths — Path from node m and n.
m is initial node in path but defining node for variable V and n is find node in path but usage node for var V.

All definitions du path
a; 2 ~~-~~ 4, 5, 8, 9, 14, 18, 19
b; 2 - 4, 5, 8, 9, 14, 18
c; 2 - 4, 5.
det; - 2 - 5, 8, 9, 19
r_1; 2 - 8, 10, 14, 15
r_2; 2 - 10, 14
rp; 2 - 20, 18
imp; 2 - 20, 19
All du path are dc-path.
Test Cases are those paths which are mentioned in du paths.