USN

CMRIT — CMR Institute of Technology, Bengaluru. Accredited with A+ Grade by NAAC

Internal Assessment Test 2 – June 2021

Sub:	Software Testing-Scheme and Answers	Sub Code:	18CS62/17 CS62	Branch:	ISE
Date:	27/06/2021	Duration: 90 min's	Max Marks: 50	Sem/Sec:	VI A,B&C

Answer any FIVE FULL Questions — MARKS | OBE: CO | RBT

1a) Define equivalence class testing. Write weak robust equivalence class test cases for commission problem. [6] CO1, CO2 L2

Definition: [2 Marks]

Test Cases [4 marks]

Definition:

- Important aspect of equivalence classes is that they form a partition of a set, where **partition refers to a collection of mutually disjoint subsets whose union is the entire set.**
- The idea of equivalence class testing is to identify test cases by using one element from each equivalence class. If the equivalence classes are chosen wisely, this greatly reduces the potential redundancy among test cases.
- **Test Cases:**

We will have **eight weak robust test cases.**

Weak Robust: Single Fault and Min- and Max+ should be considered.

Case ID	Locks	Stocks	Barrels	Expected Output
WR1	10	10	10	$100
WR2	–1	40	45	Program terminates
WR3	–2	40	45	Value of locks not in the range 1 ... 70
WR4	71	40	45	Value of locks not in the range 1 ... 70
WR5	35	–1	45	Value of stocks not in the range 1 ... 80
WR6	35	81	45	Value of stocks not in the range 1 ... 80
WR7	35	40	–1	Value of barrels not in the range 1 ... 90
WR8	35	40	91	Value of barrels not in the range 1 ... 90

1b) What are decision tables? Explain each term related to it with example. [4] CO2 L2

Decision table: Explanation [1 mark]

Example Table [1 mark]

Terms with explanation [2marks]

Stub

Rule Portion

Entry Portion

Condition Portion

- Decision tables are one of the techniques in functional testing that have been used to represent and analyze complex logical relationships
- represents a **number of combinations of actions are taken under varying sets of conditions.**
- A decision table has four portions:

1) the part to the left of the bold vertical line is the **stub portion;**

2) to the right is the **entry portion**. The part above the bold horizontal line is the

condition
portion,
3) below is the **action portion**. Thus, we can refer to the condition stub, the condition entries, the action stub, and the action entries.
4) **A column in the entry portion is a rule. Rules indicate which actions, if any, are taken for the circumstances indicated in the condition portion of the rule.**

Table 7.1 Portions of a Decision Table

Stub	*Rule 1*	*Rule 2*	*Rules 3, 4*	*Rule 5*	*Rule 6*	*Rules 7, 8*
c1	T	T	T	F	F	F
c2	T	T	F	T	T	F
c3	T	F	—	T	F	—
a1	X	X		X		
a2	X				X	
a3		X		X		
a4			X			X

2a) Define the following terms with respect to fault based testing [4] CO2 L2
i)Equivalent mutant ii) Coupling effect

Each Definition carries 2 marks.

Equivalent mutant : It is a mutant that cannot be distinguished from original program in fault based Testing
Coupling effect:
In Fault based Testing, test cases sufficient for detecting the simpler faults are sufficient also for detecting the more complex fault. This is known as the coupling effect.

2b) What is cyclomatic complexity? Write the program to find the largest of three numbers and draw the program graph. Find the cyclomatic complexity of the same. [2+2+2] CO4 L3

Cyclomatic Complexity[2 marks]
Program and Graph: [2 Marks]
Computation : [2 Marks]

Cyclomaic Complexity defines how many independent paths exists in the program or program graph. V(G)=e-n+2p e-Edges n-Nodes and p number of components in the graph.
Program:

```
void main()
{ int a, b, c, max;
print f ("Enter 3 integers");
scanf ("%d%d%d",&a,&b,&c);
if (a > b)
if (a > c)
max = a;
else
max = c;
else if (b > c)
max = b;
else max = c;
printf( "max = %d", max);
}
```

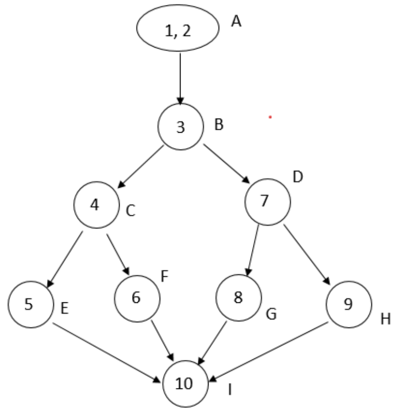
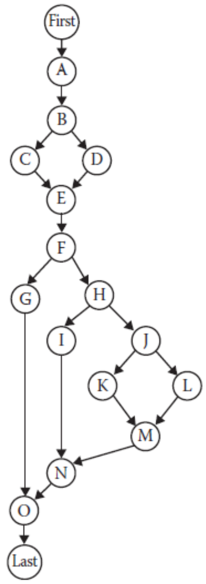
	 No. of independent paths : v (a) = e - n + 2 = 11 - 9 + 2 = 4 No. of procedures (p)= 1			
3(a)	Write notes on mutation analysis. **4 Basic points 4 marks** • Mutation analysis is the most common form of software fault-based testing. A fault model is used to produce hypothetical faulty programs by creating variants of the program under test. • Variants are created by "seeding" faults, that is, by making a small change to the program under test following a pattern in the fault model • The patterns for changing program text are **called mutation operators**, and each variant program is called a mutant. • We say a mutant is valid , if it is syntactically correct. • **We say a mutant is useful ,** if in addition to being valid, its **behavior differs from the behavior of the original program** for no more than a **small subset of program test cases**. • Mutants must be valid, **mutation operators are syntactic patterns defined relative to particular programming languages**	[4]	CO2	L2
3(b)	Define DD path graph. Draw DD path graph for triangle program problem. **DD path graph: [2 marks]** A DD-path is a sequence of nodes in a program graph such that Case 1: It consists of a single node with indeg = 0. Case 2: It consists of a single node with outdeg = 0. Case 3: It consists of a single node with indeg ≥ 2 or outdeg ≥ 2. Case 4: It consists of a single node with indeg = 1 and outdeg = 1. Case 5: It is a maximal chain of length ≥ 1. **DD path graph for triangle program problem [4 marks]**	[2+4]	CO4	L3

Figure 8.2 Nodes	DD-Path	Case of definition
4	First	1
5-8	A	5
9	B	3
10	C	4
11	D	4
12	E	3
13	F	3
14	H	3
15	I	4
16	J	3
17	K	4
18	L	4
19	M	3
20	N	3
21	G	4
22	O	3
23	Last	2



8.5 DD-path graph for triangle program.

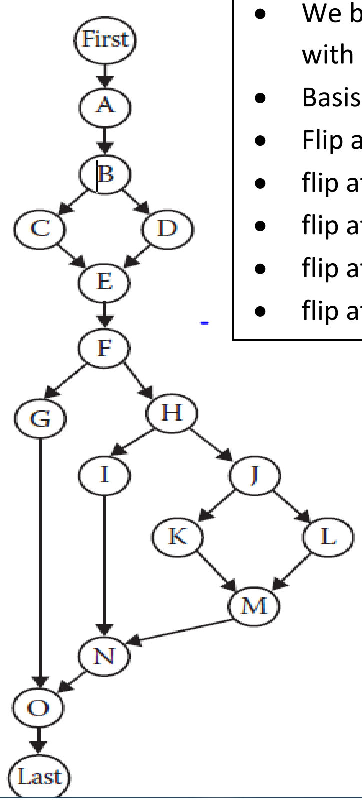
4(a) Explain McCabe's basis path testing with Triangle problem. [10] CO4 L3

Method Description [2 marks]
Program Graph and details about flipping nodes [2 marks]
Deriving basis paths [2 marks]
Removal of Feasible paths [2 marks]
Final Feasible path Computation[2 marks]

[2 marks]

- The method begins with the selection of a baseline path, which should correspond to some "normal case" program execution. This can be somewhat arbitrary;
- McCabe advises choosing a path with as many decision nodes as possible. Next, the baseline path is retraced, and in **turn each decision is "flipped"; that is, when a node of outdegree ≥ 2 is reached, a different edge must be taken.**

[2 marks]



- We begin a baseline path corresponding path with scalene Triangle.
- Basis Path: Path with highest Decision tables
- Flip at node with outdegree=2
- flip at node B
- flip at node F
- flip at node H
- flip at node J

[2 marks]

Table 8.6 Basis Paths in Figure 8.5

Original	p1: A–B–C–E–F–H–J–K–M–N–O–Last	Scalene
Flip p1 at B	p2: A–B–D–E–F–H–J–K–M–N–O–Last	Infeasible
Flip p1 at F	p3: A–B–C–E–F–G–O–Last	Infeasible
Flip p1 at H	p4: A–B–C–E–F–H–I–N–O–Last	Equilateral
Flip p1 at J	p5: A–B–C–E–F–H–J–L–M–N–O–Last	Isosceles

<u>Infeasible paths:</u> [2 marks]

- if you follow paths p2 and p3, you find that they are both infeasible.
- Path p2 is infeasible because passing through node D means the sides are not a triangle; so the outcome of the decision at node F must be node G.
- Similarly, in p3, passing through node C means the sides do form a triangle; so node G cannot be traversed.
- Paths p4 and p5 are both feasible and correspond to equilateral and isosceles triangles, respectively.
- Notice that we do not have a basis path for the NotATriangle case.
- **McCabe's procedure successfully identifies basis paths that are topologically independent;**
- however, when these contradict semantic dependencies, topologically possible paths are seen to be logically infeasible.
- One solution to **this problem is to always require that flipping a decision results in a semantically feasible path**. Another is to reason about logical dependencies
  1. If node C is traversed, then we must traverse node H.
  2. If node D is traversed, then we must traverse node G.

<u>Final Paths:</u> [2 marks]

- Taken together, these rules, in conjunction with McCabe's baseline method, will yield the following feasible basis path set. Notice that logical dependencies reduce the size of a basis set when basis paths must be feasible.

p1: A–B–C–E–F–H–J–K–M–N–O–Last	Scalene
p6: A–B–D–E–F–G–O–Last	Not a triangle
p4: A–B–C–E–F–H–I–N–O–Last	Equilateral
p5: A–B–C–E–F–H–J–L–M–N–O–Last	Isosceles

5 Consider the following C function which encodes the string in following manner. If the string character is + or - or *, it is replaced with space ' ' , if it is uppercase character, it is replaced with lowercase. Other alphanumeric characters are simply copied in destination string. Draw the control flow graph for the program. Find out the statement coverage and node coverage % from control flow graph for the following test suite T_0={"test","test**ing", "test+-"} [5+5] CO4 L3

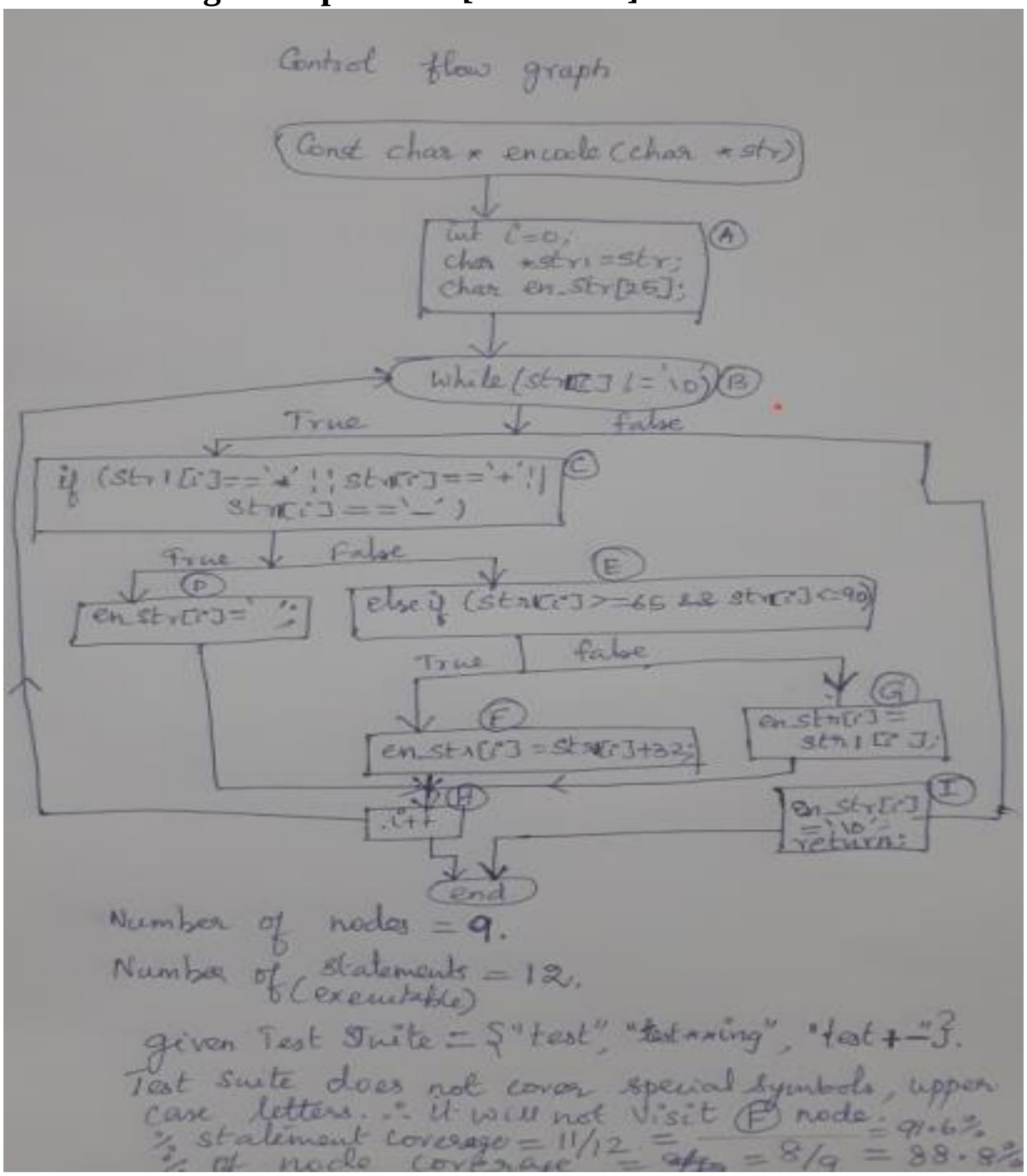
```
const char* encode(char *str) {
int i = 0;
char *str1=str;
char en_str[25];
while (str1[i] != '\0') {
if(str1[i]=='*'|| str1[i]=='+'||str1[i] =='-')
```

```
    en_str[i] =' ';
else if(str1[i]>=65 && str1[i]<=90)
    en_str[i]=str1[i]+32;
else
en_str[i]=str1[i];
i++;
}
en_str[i]='\0';
return (en_str);
}
```

Control Flow Graph: [5 marks]

Statement coverage Computation [2.5 marks]

Node Coverage Computation [2.5 marks]



6	Consider the following program. Find the DU paths for the variables staffDiscount, totalPrice, finalPrice, discount and price. Verify that whether these DU paths are definition clear.	[10]	CO4	L3

```
1  program Example()
2  var staffDiscount, totalPrice, finalPrice, discount, price
3  staffDiscount = 0.1
4  totalPrice = 0
5  input(price)
6  while(price != -1) do
7      totalPrice = totalPrice + price
8      input(price)
9  od
10 print("Total price: " + totalPrice)
11 if(totalPrice > 15.00) then
12     discount = (staffDiscount * totalPrice) + 0.50
13 else
14     discount = staffDiscount * totalPrice
15 fi
16 print("Discount: " + discount)
17 finalPrice = totalPrice - discount
```

20 DU paths * 0.5 marks = 10 marks

A definition/use path with respect to a variable v **(denoted du-path)** is a path in PATHS(P) such that, for some v ∈ V, there are define and usage nodes DEF(v, m) and USE(v, n) such that m and n **are the initial and final nodes of the path**

DU Path for staffDiscount

P1(3, 12) = <3,4,5,6,7,8,9,10,11,12> - is definition clear

P2(3, 14) = <3, 4,5,6,7,8,9,10,11,12,13,14> **is NOT** definition clear

DU Path for totalPrice

P3(4, 7) = <4,5,6,7 > - definition clear

P4(4, 10) = <4,5,6,7,8,9,10> **NOT** definition clear

P5(7, 6) = <7,8,9,6> definition clear

P6(7,7) = <7,8,9,6,7> **NOT** definition clear

P8 (7, 10) = <7,8,9,6,10> definition clear

P9(7, 11) = <7,8,9,6,10,11> definition clear

P10(7, 12) = <7,8,9,6,10,11,12> definition clear

P11(7, 14) = <7,8,9,6,10,11,12,13,14> definition clear

DU Path for finalPrice

P12(17,17) = <17,17> - definition clear

DU Path for discount

P13(12,16)=<12,13,14,15,16> > **NOT** definition clear

P14(12,17)=<12,13,14,15,16,17> **NOT** definition clear

P15(12,16)=<12,13,14,15,16> > **NOT** definition clear

P16(14,16)=<14,15,16> > definition clear

P17(14,17)=<14,15,16,17> > definition clear

DU Path for price

P18(5,6) =<5,6> definition clear

P19(5,7)=<5,6,7> definition clear

P20(8,6)=<8,9,6> definition clear

P20(8,7)=<8,9,6,7> definition clear