BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJASTHAN)

First Semester, 2023-2024

45 Marks (45% Weightage) **Comprehensive Examination** **Closed Book**

Course Number: SS G552 **Course Title:** Software Testing Methods

Date : December 11, 2023 **Time** : 02:00 PM – 05:00 PM (AN)

Note: There are seven questions in all. Write your answers to the point. **All your answers to the questions must be supported by reason(s) or appropriate justification or steps to solve the problem as applicable.**

Q1. Answer the following:

1.1. In the *Table Q1*, match the entries in Column A with two appropriate entries in Column B. Write your answer as *"a. -> X, Y"* where *X* and Y represents the two matching labels (among A, B, C ... K, L) of the item-name from the Column B. Note that an answer will be considered correct only when it is correct with respect to the completely correct matching (solution).

1.2. Find the cyclomatic complexity of the CFG *[(A, B), (A,C), (B,D), (B,E), (C,E), (C, F)]*, given that the node A is the start node.

1.3. Find one basis path passing through the node 5 for the CFG *[(Start, 1), (1,2), (1,3), (2,4), (2,5), (3, 5), (3, 6), (4, 7), (5,7), (6,7), (7, End)]*.

Column A	Column B
a. Path testing	A. Statement coverage
b. Control flow graph	B. Nested
c. Path testing criteria	C. Concatenated
d. Loops	D. Branch coverage
e. Predicate	E. Structural testing
f. Data Flow testing	F. Decision
	G. Simple
	H. define-use path
	I. Compound
	J. Usage node of a variable
	K. Mostly for unit testing
	L. Process block

Table Q1

Marks Q1 [(0.75 x 6 = 4.5) + 1.5 + 1.5 = 7.5]

Q2. For the program given in the Table Q2a, discuss about the correctness of the program iteration structure with respect to the method described in the Table Q2b.

Marks Q2 [7.5]

Suppose b is a Boolean expression, S is a sequence of statements, P is Precondition and Q is Postcondition, then the correctness of an iteration structure is shown as P { while b do S} Q. A predicate that meets the following conditions is called a *loop invariant*:

1. Predicate is True before evaluating the loop condition.
2. Predicate is True at the start of the initial iteration of the loop.
3. If the loop invariant and the loop condition are True at the start of any iteration, then the invariant is True at the end of that iteration, and
4. Conjunction of the loop variant and the negation of the loop condition implies Q.

Whenever there is a loop invariant for an iteration structure, the correctness of the structure can be shown using some form of induction. If a predicate is found that satisfies the four properties of the loop invariant predicate, then the iteration structure is called a *partially correct iteration*. If it can be shown that the iteration will always terminate, then the iteration structure is called a *totally correct iteration*.

In order to show that a loop will terminate, we need to define a *decrementing function* associated with the loop structure and show that in each iteration the decrementing function decreases. A decrementing function is mapped into a set of descending positive integers ending with zero.

Table Q2b

Precondition is:
P = (Sum = 0 and I = 0 and N >= 1)
Program is:
{ while I < N do
Begin
I = I + 1;
Sum = Sum + I
End; }
Postcondition is:
Q = (Sum = 1 + 2 + 3 + … + I)

Table Q2a

Q3. A path between two nodes (node-names represented by lowercase characters) in a CFG is also called as *path-name*. Any expression which denotes a set of path-names (not necessarily the set of all paths) between two nodes is called a *path expression*. We denote set of paths by uppercase letters such X, or Y etc. The name of a path that consists of two successive path segments is conveniently expressed by the concatenation or *path product* of the segment names. The

path sum denotes paths in parallel between two nodes. If X and Y are sets of paths that lie between the same pair of nodes, then X + Y denotes the union of those sets of paths. Answer the following:

3.1 Why the expression derived from path expressions may be commutative, even though the path product is not commutative?

3.2 Are the both path-product and path-sum operations distributive?

3.3 Why the path sum is both commutative and associative?

3.4 Is the Absorption rule valid for path-sum or path-product or both?

3.5 Why is it true that any arbitrary sum identical path expressions reduce to the same path expression?

Marks Q3 [1 x 5 = 5]

Q4. Given a date in the format dd, mm, yyyy where dd, mm, yyyy are positive valid integers, the NextDate function finds the next date. Do the following:

4.1 Write equivalence classes if the year ranges in [1812, 2012].

4.2 Write at least five test cases based on the equivalence classes derived above.

Marks Q4 [2.5 x 2 = 5]

Q5. Answer w.r.t. an FSM, with initial state q_1, as given in the Table Q5.

5.1 Draw the FSM.

5.2 Draw FSM when Reset inputs are used.

5.3 Is the FSM completely specified?

5.4 Is the FSM strongly connected?

Current State	Output when input is		Next State when input is	
	a	b	a	b
q_1	0	1	q_1	q_4
q_2	0	1	q_1	q_5
q_3	0	1	q_5	q_1
q_4	1	1	q_3	q_4
q_5	1	1	q_2	q_5

Table Q5

Marks Q5 [1.25 x 4 = 5]

Q6. For the functional decomposition shown in Figure Q7, answer the following:

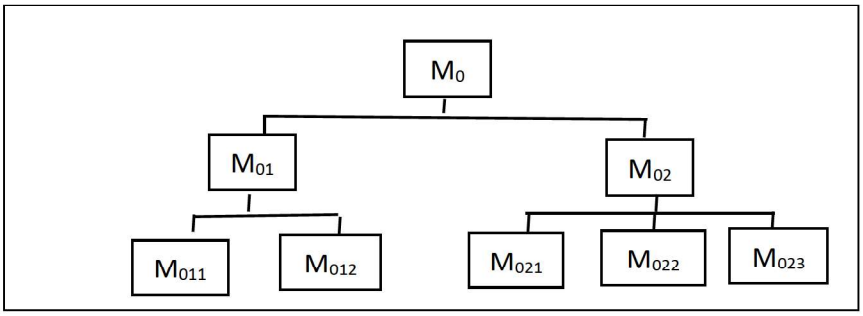

Figure Q6

6.1 Calculate the number of steps needed for bottom-up integration testing. Write the integration to be tested in each step in correct order.

6.2 Calculate the count of maximum number of stubs needed? For which modules these stubs may be needed.

6.3 Calculate the count of maximum number of drivers needed? Write for which modules these drivers may be needed.

6.4 How many sequences are needed for Sandwich integration using Big-Bang? Write the sequences in correct order.

6.5 How many sequences are needed for Sandwich integration using Bottom-up approach? Write the sequences in correct order.

Marks Q6 [1.5 x 5 = 7.5]

Q7. Answer the following briefly.

7.1 For OO software testing write two advantages of choosing class-as-unit?

7.2 For OO software testing what are the implications of using composition as central design strategy?

7.3 What is the essence of system-level testing for GUI applications?

7.4 Write two fundamental limitations of the functional testing.

7.5 Give an example of each of 'very serious fault' and 'extreme fault' with respect to a transaction processing system.

Marks Q7 [1.5 x 5 = 7.5]