Daffodil International University
Faculty of Science & Information Technology
Department of Software Engineering
Final Examination, Summer- 2026
Course Code: SE 312
Course Title: Software Quality Assurance & Testing
Sections & Teachers: (A-B, SA) (C-AUA) (D-E-QFF) (F, AJE) (G,FM) (H,I,J-SIM)
Time: 2 Hours Marks: 40

Answer ALL Questions
[The figures in the right margin indicate the full marks and corresponding course outcomes. All portions of each question must be answered sequentially.]

1	a)	A university library allows students to borrow books under the following conditions: • A student can borrow a maximum of **5 books**. • Books already issued cannot be borrowed again. • Students with overdue books cannot borrow new books. • Only active student accounts are allowed to borrow books. • Borrowing is valid for **14 days**. **Illustrate** Five (5) test cases to verify the book borrowing system.	[Marks-5]	CLO-3 Level-3

	b)	**Code-1**
		#include <stdio.h>
		
		int calculateTotal(int marks) {
		1. int i, total = 0;
		2. for(i = 1; i <= marks; i++)
		3. {
		4. total = total + i;
		5. }
		6. return total;
		7. }
		
		int main() {
		1. int marks, totalMarks;
		2. printf("Enter a positive number: ");
		3. scanf("%d", &marks);
		4. if(marks <= 0)
		5. {
		6. printf("Invalid Input\n");
			[Marks-4+2+4]	CLO-3 Level-3

```
7.  }
8.  else
9.  {
10.     totalMarks = calculateTotal(marks);
11.     if(totalMarks >= 100)
12.       printf("Excellent\n");
13.     else if(totalMarks >= 50)
14.       printf("Good\n");
15.     else
16.       printf("Needs Improvement\n");
17.     printf("Total = %d\n", totalMarks);
18. }
19. printf("Program Completed\n");

20. return 0;
21. }
```

From Code-1,

a. **Draw** the Control Flow Graph (CFG) for the given program.

b. **Determine** any two independent paths.

Code-2

```
#include <stdio.h>
int main()
{
1.  int score, choice;

2.  scanf("%d", &score);
3.  scanf("%d", &choice);

4.  while(score > 0)
5.  {
6.    if(score >= 80)
7.      printf("A\n");
8.    else if(score >= 50)
9.      printf("B\n");
10.   else
11.     printf("C\n");

12.   switch(choice)
13.   {
14.     case 1:
15.       printf("Red\n");
16.       break;

17.     case 2:
18.       printf("Blue\n");
19.       break;

20.     case 3:
```

```
21.            printf("Green\n");
22.            break;

23.         case 4:
24.            printf("Yellow\n");
25.            break;

26.         case 5:
27.            printf("Black\n");
28.            break;

29.         default:
30.            printf("Invalid\n");
31.      }
32.   score -= 25;
33.   }
34.   return 0;
35. }
```

c. **Calculate** the Cyclomatic Complexity of the program from Code-2. Do not need to draw the CFG.

	c)	While testing an Online Banking System, a tester discovers that users cannot transfer money when the transfer amount contains decimal values (e.g., 1000.50). The tester reports the defect, the developer fixes it, and the tester successfully verifies the fix. I. **Draw** a simple Bug Life Cycle diagram to illustrate the process followed in this scenario. II. Why was the bug **Reopened** after it had been marked as **Resolved**?	[Marks-5]	CLO-3 Level-3

2	a)	`#include <stdio.h>`
		`void process(int age, int marks, int attendance) {`
		`  if (age >= 18 && attendance >= 75)`
		`  printf("Eligible\n");`
		`  else`
		`  printf("Not Eligible\n");`
		`  if (marks >= 80 || attendance >= 90)`
		`  printf("Scholarship\n");`
		`  else`
		`  printf("No Scholarship\n");`
		`  printf("Completed\n");`
		`}`
		
		Design the minimum number of test cases required to achieve 100% **Statement Coverage, Branch Coverage, Decision Coverage, and Condition Coverage**. For each coverage technique:
		➢ Create the minimum set of test cases.
		➢ **Calculate** and show the coverage percentage after each test case.
			[Marks-6]	CLO-4
				Level-3

b)	Consider the following C program (0-based indexing):
	
	`int analyze(int a[4]) {`
	`  int count = 0, sum = 0;`
	`  for (int i = 0; i < 4; i++) {`
	`  if (a[i] > 0) {`
	`  if (a[i] % 2 == 0)`
	`  count++;`
	`  else`
	`  sum += a[i];`
	`  }`
	`  }`
	`  if (count >= 2)`
	`  return sum + count;`
	`  else`
	`  return sum - count;`
	`}`
	The following mutants are generated.
	• M1: if (a[i] >= 0),
	• M2: for (int i = 0; i < 4; i += 2),
	• M3: count += 2;
	• M4: if (count > 2)
	Use the following test cases:
	• T1 = {2, 3, 4, 5}
	• T2 = {0, -1, -2, -3}
	I. **Simulate** the original program for test cases T1 and T2. Show the values of count, sum, and the final returned output.
	II. **Simulate** mutants M1–M4 using both test cases. Compare each mutant's final output with the original program and identify whether each mutant is killed or survives
	III. Using test case T1, analyse mutant M3 according to the RIP model: Reachability, Infection, and Propagation and determine whether it is strongly killed.
	IV. **Calculate** the mutation score for the complete test suite {T1, T2}.
		[Marks-2+3+2+1=8]	CLO-4 Level-3

c)	A QA team is testing a newly developed Hospital Management System containing 8,000 Lines of Code (LOC). During initial testing, the team identified 300 defects. In the first fixing cycle, developers worked for 30 business days and fixed 240 defects. After re-testing, QA found that 45 of the fixed defects were not properly resolved and rejected them. In the second fixing cycle, developers resolved 50 defects within 15 business days. During verification, QA rejected 10 defects due to incomplete or incorrect fixes. Based on the above scenario, **Calculate** the following: **i. Defect Density** **ii. Defect Removal Rate (DRR)** **iii. Defect Rejection Rate for each fixing cycle** **iv. Turn Around Time (TAT) for each fixing cycle**	[Marks-6]	CLO-4 Level-4