National University of Computer and Emerging Sciences

Lahore Campus

0130

70

Soft. Qty. Eng. (SE3002)

Final Exam

Date: December 18th 2024

Course Instructor(s)

Dr. Ali Afzal Malik (BSE-5A; BSE-5B)

Ms. Javeria Sadiq (BSE-5C)

Sections: ALL

Total Time: 3 Hours

Total Marks: 110

Weight: 40%

Total Questions: 05

[redacted]

Student Name | Roll No | Section

Instructions: Attempt all questions on the question paper. Neither use nor submit any extra sheet. Usage of calculators is allowed.

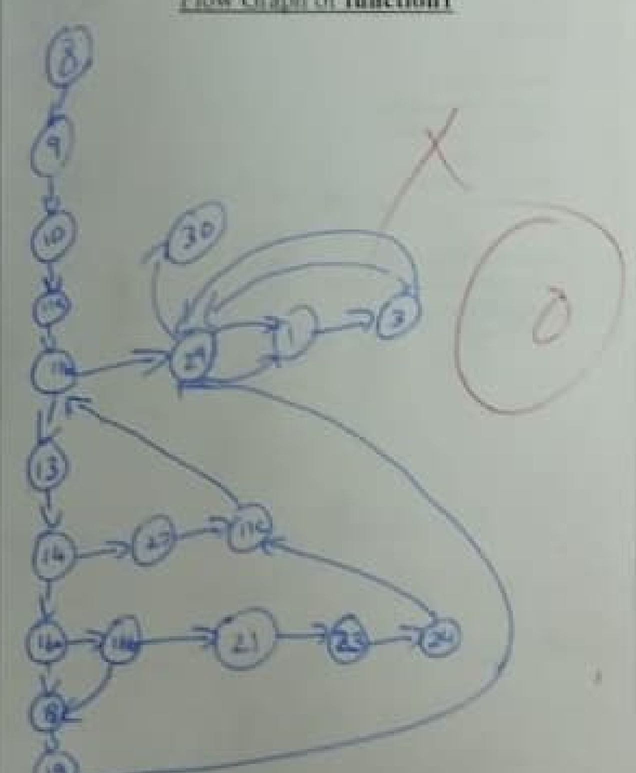
CLO 3: Understand the use of software quality related metrics in Software Quality Engineering

Q1 .... [10 + 10 + 3 + 3 + 4 = 30]

a. Draw the flow graph of **function1** given below inside the box appearing besides its code. Nodes must be annotated clearly on the code. [Note: Line numbers have been added at the start of each line of code for convenience only.]

```
1. int square(int num)
2. {
3.    return (num * num);
4. }
5.
6. int function1 (int a, int b)
7. {
8.    int c = 1;
9.    int d = 1;
10.   int e;
11.   for(int i = 0; i < a; i++)
12.   {
13.      c++;
14.      while(c < b)
15.      {
16.         if((a == b) || (c > 100))
17.         {
18.            d++;
19.            break;
20.         }
21.         else
22.         {
23.            d--;
24.            continue;
25.         }
26.      }
27.      cout<<"Iteration = "<<c<<"\n";
28.   }
29.   e = square(c) + square (d);
30.   return e;
31. }
```

Flow Graph of **function1**



0

c. What is the maximum number of linearly independent paths in the flow graph of **function1**? Show your working clearly.

8 → 9 → 10 → 11a → 11b → 13 → 14 → 16a → 16b → 18 → 19 → 29 → 1 → 3 → 29 → 30

8 → 9 → 10 → 11a → 11b → 13 → 14 → 16a → 16b → 21 → 23 → 24 → 11c → 11b → 19 → 1 → 23 → 29 → 30

8 → 9 → 10 → 11a → 11b → 13 → 14 → 16a → 18 → 19 → 21 → 23 → 29 → 30

8 → 9 → 10 → 11a → 11b → 13 → 14 → 22 → 11c → 11b → 29 → 1 → 3 → 29 → 30

24 is maximum no. of linearly independent paths as shown by dashes.

d. What is the maximum number of linearly independent paths in the flow graph of **function2**? Show your working clearly.

13 → 14 → 16 → 17 → 23 → 24 → 26 → 30 → 33 → 34 → 35

13 → 14 → 20 → 23 → 24 → 30 → 33 → 34 → 35

13 → 14 → 19 → 1 → 8 → 20 → 23 → 24 → 28 → 6 → 8 → 28 → 29 → 33 → 34 → 35

13 → 14 → 18 → 1 → 3 → 18 → 20 → 23 → 24 → 27 → 1 → 3 → 27 → 30 → 33 → 34 → 35

Hence, 53 is maximum no. of linearly independent paths.

e. Which of these two functions (function1 or function2) is likely to have more errors? Why?

Function likely to have more errors: function2

Reason:

Let's calculate cyclomatic complexity:-

for function 1:-

$V(G) = E - N + 2 = 26 - 20 + 2 = 8$

$V(G) = P + 1 = 6 + 1 = 7$

$V(G) = R = 7$

Since 7 is more freq., so $V(G)$ for func1 = 7.

for function 2:-

$V(G) = E - N + 2 = 33 - 21 + 2 = 14$

$V(G) = P + 1 = 8 + 1 = 9$

$V(G) = R = 9$

Since 9 is more freq., so $V(G)$ for func2 = 9

Now, $V(G)_2 > V(G)_1$, which means that project 2 is like to have more errors, More project 2 has more lines of code. when it is checked for errors, it will cause less branch coverage and will have more errors as well.

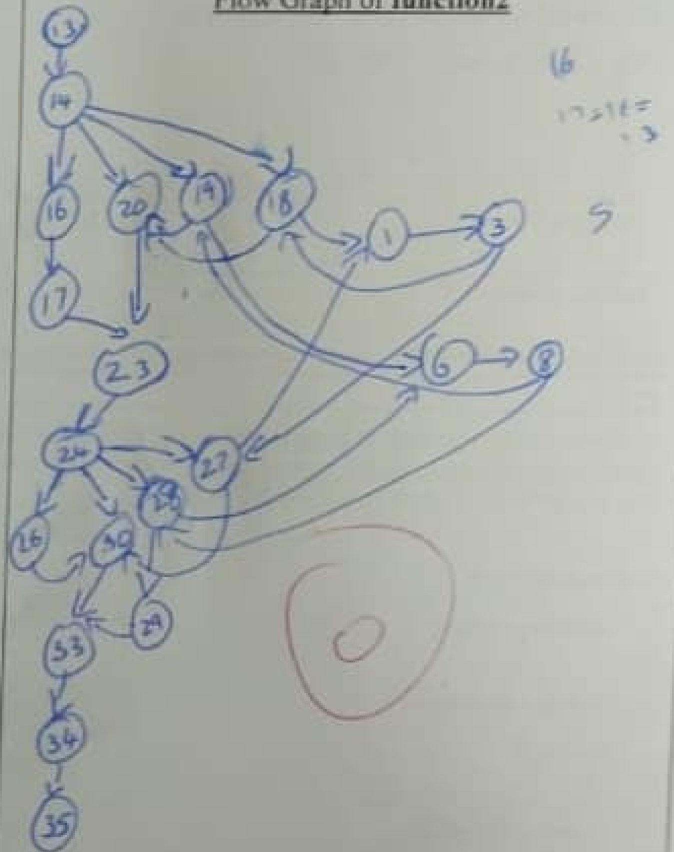
4

b. Draw the flow graph of **function2** given below inside the box appearing besides its code. Nodes must be annotated clearly on the code. [Note: Line numbers have been added at the start of each line of code for convenience only.]

```
1. int square(int num)
2. {
3.   return (num * num);
4. }
5.
6. int cube(int num)
7. {
8.   return (num * num * num);
9. }
10.
11. int function2 (int a)
12. {
13.   int b;
14.   switch(a)
15.   {
16.     case 1: b = a;
17.             break;
18.     case 2: b = square(a);
19.     case 3: b = cube(a);
20.     default:   b = 0;
21.
22.   }
23.   int c;
24.   switch(b)
25.   {
26.     case 1: c = b;
27.     case 4: c = square(b);
28.     case 8: c = cube(b);
29.             break;
30.     default:   c = 0;
31.
32.   }
33.   cout<<"b = "<<b<<"\n";
34.   cout<<"c = "<<c<<"\n";
35.   return (b + c);
36. }
```

Nodes ?

Flow Graph of function2



4

CLO 3: Understand the use of software quality related metrics in Software Quality Engineering

Q2 .... **[8 + 2 = 10]**

The developers of FASTTravel – an airline, hotel, and taxi booking web app for frequent travelers – have decided to employ the dual independent testing teams method. They determined their testing termination level to be residual undetected errors of 0.85%. $P(a)(b) = 0.85$

After 3 weeks of testing, testing team A has detected 390 (N_a) errors while testing team B has detected (N_b) 380 errors. Comparison of the detected errors of the two teams revealed that 350 errors (N_{ab}) had been detected by both testing teams.

a. Using the information given above, estimate each of the following (use **3** decimal places):

i. Proportion of errors detected by team A

$P_a = \frac{N_a}{N_{ab}} = \frac{390}{350} = 1.11$ 0.921

ii. Proportion of errors detected by team B

$P_b = \frac{N_b}{N_{ab}} = \frac{380}{350} = 1.08$ 0.897

2

iii. Total number of errors in FASTTravel

$N = \frac{N_a \times N_b}{N_{ab}} = \frac{390 \times 380}{350} = 423$. 429 errors

iv. Proportion of errors undetected by both teams

$P(a)(b) = (1-P_a)(1-P_b)$

$= 0.009$ 0.81.

b. Should testing be terminated at this stage? Justify your answer.

Decision: Continue

Justification:

Since $P(a)(b) = 0.009 >$ undetected errors $= 0.0085$ we should continue testing, until this undetected proportion is reduced.

Moreover, the proportion has increased instead of decreasing.

National University of Computer and Emerging Sciences
Lahore Campus

12

CLO 3: Understand the use of software quality related metrics in Software Quality Engineering

Q3 .... [2 + 2 + 4 + 4 + 2 + 2 + 2 + 2 = 20]

FASTComplain is a web-based complaint management system developed using 45,000 lines of all new JavaScript code for managing the complaints reported by the community of FAST-NUCES. A team of 10 software engineers each working for 40 hours a week for 3 weeks developed FASTComplain in three phases. In the first phase, the features relevant to faculty were developed. Then, the features relevant to staff were developed. Finally, the features relevant to students were developed. The end of each phase corresponded to a project milestone. Only the first milestone was completed on time. The rest of the milestones were completed after a delay of 10 days and 5 days for milestone 2 and milestone 3, respectively. During the development of FASTComplain only two SQA activities were used to detect errors i.e. design reviews and basis path testing. Design reviews detected 5 design-related errors and basis path testing uncovered 25 code-related errors. No other errors were detected. After performing alpha and beta testing, FASTComplain was released on January 1, 2015. From January 1, 2015 to June 30, 2015 (both dates included), 15 help desk calls were received and 3 failures were detected and reported by the FAST-NUCES community. From July 1, 2015 to December 31, 2015 (both dates included), 20 help desk calls were received and 4 failures were detected and reported by the FAST-NUCES community. The maintenance team spent a total of 43 hours to correct these failures detected and reported by the FAST-NUCES community in 2015. From January 1, 2015 to December 31, 2015 (both dates included), FASTComplain was up and running 24 hours a day on all days except August 14, 2015. On August 14, 2015 the server hosting FASTComplain was switched off the entire day for performing upgrades and archiving data.

Without making any assumptions, use only the information provided above to calculate the following metrics for FASTComplain. Show complete working (including formulas used). Answers (values) must be correct to three decimal places. <u>There is no partial credit.</u>

#	Metric	Value
1	Time Table Observance (TTO)	Milestones Completed On Time (MSOT) / No of Milestones (MS) = 1/3 = 0.333 milestones
2	Average Delay of Milestone Completion (ADMC)	Total Milestones Delay At completion (TMDAC) / No. of Milestones (MS) = (10+5)/3 = 15/3 = 5 days/milestone
3	Development Error Removal Effectiveness (DERE) NDE = 5+25 = 30	No. of Development Errors (NDE) / NDE + No. of failures in year (NYF) = 30/(30+~~3~~7) = 30/37 = 0.8~~11~~ errors
4	Development Productivity (DevP)	No. of hrs. for Dev. (DevH) / Modified code in 10^3 (KLMC) = 40x3/45 = 120/45 = 2.667 hrs 26.667
5	HD calls Density (HDD)	No. of HD Calls (NHYC) / Mod. code in 10^3 (KLMC) = 15/45 = 0.333 calls 0.778
6	Software System Failure Density (SSFD)	No. of failures in Year (NYF) / Mod. Code in 10^3 (KLMC) = 34/45 = 0.156 failures
7	Corrective Maintenance Effectiveness (CMaiE)	No. of failures / No. of hours in maintenence = 7/43 = 0.163 failures 6.143
8	Total Unavailability (TUA)	No. of hrs total service is off yearly (NYTFH) / No. of Hours Service runs yearly (NYSerH) = 24/(24x364) = 0.003 hrs

CLO 3: Understand the use of software quality related metrics in Software Quality Engineering

Q4 .... [4 + 8 + 8 = 20]

The following table shows a partially complete Inspection Session Summary Report. Complete this Report by filling in the missing information. Use **two** decimal places for values of defect detection metrics. Check your answers carefully. There is **no** credit for errors carried forward.

Inspection Session Summary Report

Session date: February 1, 2020
The inspected document: Software Project Plan (SPP)
The inspected document sections: All
The inspection team: Bo (inspection leader), Fo, Ho, Jo, Mo

Project Name: FASTFood
Version: 3
Total Pages: 50

1 Resources invested (hours worked) (Max. Marks = 4)

#	Team member	Overview meeting	Preparation	Inspection session	Total (hours)	Comments
1	Bo	2	7	3	12	Including report preparation
2	Fo	2	6	2	10	
3	Ho	2	5	2	9	
4	Jo	2	4	2	8	
5	Mo	2	5	2	9	
	Total	10	27	11	48	

2 Defect summary (Max. Marks = 8)

Defect severity	Defect nature			Total defects	Severity factor	Total defects (standardized)	Comments
	W	M	E*				
5 – critical	2	2	1	5	16	80	
4	5	1		6	8	48	
3		4	2	6	4	24	
2	5		3	8	2	16	
1 – minor	10	5	1	16	1	16	
Total	22	12	7	41		184	

3 Defect detection metrics (Max. Marks = 8)

(1) Average defects per page = $\frac{\text{No. of defects}}{\text{No. of pages}} = 41/50 = 0.82$

(2) Standardized average defects per page = $\frac{\text{No. of Stand. def.}}{\text{No. of pages}} = \frac{184}{50} = 3.68$

(3) Defects detection efficiency (hours per defect) = $\frac{\text{Total hrs}}{\text{No. of defects}} = 48/41 = 1.17$

(4) Standardized defect detection efficiency = $\frac{\text{Total hrs}}{\text{No. of Stand. def.}} = 48/184 = 0.261$

Prepared by: Bo.
Signature: Bo
Date: February 3, 2020

*W = Wrong; M = Missing; E = Extra

20

30

CLO 3: Understand the use of software quality related metrics in Software Quality Engineering

Q5 .... [20 + 8 + 2 = 30]

SatelliteSoft, a local software house which specializes in developing software components for satellites, is considering adopting one of the following two SQA plans (see Table 1 and Table 2) for its SatFast project.

Table 1: SQA Plan 1

No.	SQA Activity	Defect Removal Effectiveness	Cost of Removing a Detected Defect (Cost Units)
1	Requirements specification review	60%	1
2	Design review	60%	2.5
3	Code inspection	70%	6.5
4	Integration test	60%	16
5	System test	60%	40
6	Operation phase	100%	110

Table 2: SQA Plan 2

No.	SQA Activity	Defect Removal Effectiveness	Cost of Removing a Detected Defect (Cost Units)
1	Requirements specification review	60%	1
2	Design inspection	70%	2.5
3	Unit test - code	50%	6.5
4	Integration test	60%	16
5	System test	60%	40
6	Operation phase	100%	110

Table 3: Distribution of Software Defect Origins

No.	Software development phase	Average percentage of defects originating in phase
1	Requirements specification	25%
2	Design	35%
3	Coding (coding 30%, integration 10%)	40%

There is **no** partial credit in this question and there are **no** marks for errors carried forward. Therefore, check your answers carefully. All numerical values should be accurate to **3 decimal places.**

a. Using the defect origin distributions in Table 3, complete the process-oriented illustrations of SQA Plan 1 and SQA Plan 2 models given below assuming the removal of 100 defects.