Name: _____ Campus ID:_____

Time Allowed: 60 min Total Marks: 40

CS-360 Software Engineering

Quiz - 4

Question 1. Choose the correct option. Give proper Rational for your choice. (No credit without rational). (5x1=5)

1. Which of the followings is FALSE about Multiple Condition Coverage (MCC):
    a) Every point of entry and exit in program has been invoked at least once.
    b) Every condition in a decision in program has taken all possible outcomes at least once.
    c) Every statement in the program has been involved at least once.
    d) Every condition in a decision has been shown to independently affect that decision's outcome.
    e) None of these

Rational:

2. Adding a mouse hover animation on a button in a website supports which of the following design principles:
    a) Affordance
    b) Visibility
    c) Feedback
    d) Mapping

Rational:

3. Which of the following breaks internal consistency:
    a) Using different design as compared to other websites within same industry.
    b) Using different headers, footers, and fonts across multiple websites of same company.
    c) Using different "checkout" processes as compared to other online stores.
    d) None of the above

Rational:

4. Replacing a simple text box to enter a city name with a drop-down menu (listing city names) follows which of the following design principles:
    a) Constraint
    b) Visibility
    c) Feedback

d) Mapping

Rational:

5. In the early generations of computing, what was the primary reason for using waterfall methodology?

e) Code Base Size
f) Cost of Change
g) Lack of developers
h) Extensive documentation

Rational:

Question 2. Mark statements as True or False. (5x1=5)

1. In condition coverage, every statement in the program has been invoked at least once. **T / F**
2. In path coverage, the number of paths is exponential to number of branches. **T / F**
3. Predicate coverage only includes path coverage. **T / F**
4. Path coverage includes decision coverage. **T / F**
5. MCC ignores branches within Boolean expressions which occur due to short-circuit operators. **T / F**

Question 3. Short Questions (3+4+3=10)

1. "Highest level of testing productivity occurs when we identify the most failures with the least effort." Justify the statement with an example.

Effort is measured by the time required to create test cases, add them to your test suite and run them. It follows that you should use a coverage analysis strategy that increases coverage as fast as possible. This gives you the greatest probability of finding failures sooner rather than later. One strategy that usually increases coverage quickly is to first attain some coverage throughout the entire test program before striving for high coverage in any particular area. By briefly visiting each of the test program features, you are likely to find obvious or gross failures early. For example, suppose your application prints several types of documents, and a bug exists which completely prevents printing one (and only one) of the document types. If you first try printing one document of each type, you probably find this bug sooner than if you thoroughly test each document type one at a time by printing many documents of that type before moving on to the next type. The idea is to first look for failures that are easily found by minimal testing.

2. Consider the following C++ code fragment:

```
bool isEquiTriangle(int a, int b, int c) {
    if (a <= 0 || b <= 0 || c <= 0) {
        return false;
    }
    if (a + b <= c || a + c <= b || b + c <= a) {
        return false;
    }
    if (a == b && b == c) {
        return true;
    }
    return false;
}
```

Write test cases that achieve Modified Condition/Decision Coverage (MC/DC) for the isEquiTriangle function.

Cover all conditions within each decision, and combinations of all conditions.

- 8 test cases of first IF condition:
  Only a<=0, Only b<=0, Only c<=0, a<=0 and b<=0, b<=0 and c<=0, a<=0 and c<=0, a<=0 and b<=0 and c<=0, a>0 and b>0 and c>0)
  E.g., (-1, 2, 3), (1,-1,2), (1,2,-2), (-1,-2,1), (1,-2,-3),(-1,2,-3),(-1,-2,-1),(1,2,3)

- similarly, 8 test cases for second IF condition.
- 4 test cases for thirs if condition. (only a==b, only b==c, a==b and b==c, a!=b and b!=c).

Marking scheme:

Total test cases = 20
0.2 marks/test case.
e.g., if a student gives 4 test cases only, he'll get 4*0.2=0.8 marks

3. According to Grady Booch, what changes to the software led to the adoption of agile software development life cycle?

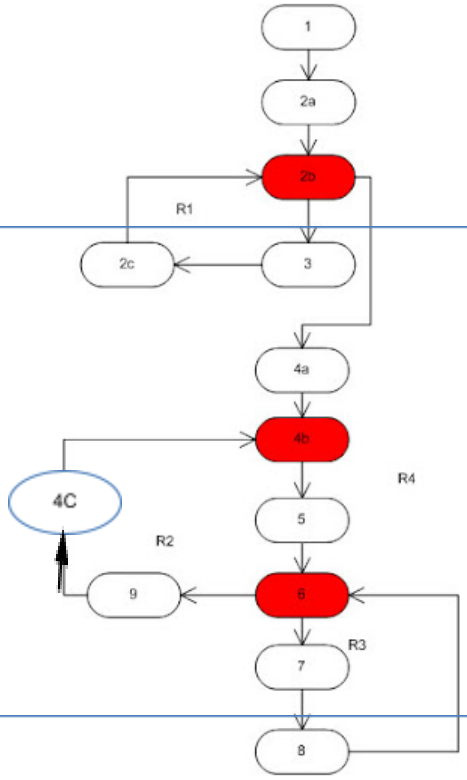
   Answer
   With the increase of code size, the issue of legacy systems occurred. Using agile methodology, we can create stable states of the software and easily revert back to make changes.
   Marking Scheme:
   - If the student discusses about code size and legacy systems give them full marks

Question 4. Consider the following C++ code fragment:

```
insertion_procedure (int a[], int p [], int N)
{
int i,j,k;
for (i=0; i<=N; i++)
{
    p[i] = i;
}
for (i=2; i<=N; i++)
{
       k = p[i];
       j = 1;
       while (a[p[j-1]] > a[k])
```

```
            {
                    p[j] = p[j-1];
                     J-
            }
            p[j] = k;
      }
      }
```

a) Draw the flow graph. (3 marks)



Start by numbering the statements:

```
 insertion_procedure (int a[], int p [], int N)
 {

1)    Int i,j,k;
(2)    for ((2a)i=0; (2b)i<=N; (2c)i++)
(3)        p[i] = i;
(4)    for ((4a)i=2; (4b)i<=N; (4c)i++)
       {
(5)       k=p[i];j=1;
(6)       while (a[p[j-1]] > a[k]) {
(7)           p[j] = p[j-1];
(8)           j--
          }
(9)     p[j] = k;
   }
}
```

b) Calculate the cyclomatic complexity (using all 3 formulas). (3 marks)

V(G) = Number of simple predicates (red on graph) + 1 = 3 + 1 = 4
V(G) = E - N+2 (where E are edges and N are nodes) = 15 - 13 + 2 = 4
V(G) = Number of enclosed areas + 1 = 3 + 1 = 4

c) List the basis paths. (3 marks)

To list all the basis paths in the given code, we need to identify all possible paths through the code that cover each statement at least once. Here are the basis paths for the given code:

1) 1, 2a, 2b, 4a, 4b
2) 1, 2a, 2b, 4a, 4b, 5, 6, 9, 4c, 4b
3) 1, 2a, 2b, 4a, 4b, 5, 6, 7, 8, 6, 9, 4c, 4b
4) 1, 2a, 2b, 4a, 4b, 4c, 5, 6, 7, 8, 9
5) 1, 2a, 2b, 3, 2c, 4a, 4b
6) 1, 2a, 2b, 3, 2c, 4a, 4b, 5, 6, 9, 4c, 4b
7) 1, 2a, 2b, 3, 2c, 4a, 4b, 5, 6, 7, 8, 6, 9, 4c, 4b
8) 1, 2a, 2b, 3, 2c, 4a, 4b, 4c, 5, 6, 7, 8, 9

d) Prepare a test case for one path. (1 marks)

A=[1,2,3,4] B=[5,6,7,8], N=0
Testing path 1.

Question 5. Consider the following C++ code snippet:

```
void reverseArray(int arr[], int size) {
    int temp[size]; // temporary array
    int i = 0;
    while (i < size) {
        if (i=1)
        {
           continue
        }
        temp[size - 1 - i] = arr[i];
        i++;
    }
    i = 0;
    while (i < size) {
        arr[i] = temp[i];
        i++;
    }
}
```

a) Perform Syntactic Analysis. (2 marks)

Identify errors in syntax.

- == instead of =
- ; missing

b) Perform Data Analysis and Control Flow Analysis. (3 marks)

Create a flow diagram.
Data Analysis:
Mark each data object in diagram as definition (D), usage (U) and elimination (E).
Analysis (involves flow graph traversal, e.g. DD paths suggest redundancy, DE paths are most likely to be bugs.)
Control Flow Analysis:
Identify dead code.
Nonterminating decisions.
Poor structure.

c) Perform Program Slicing (Forwards and Backwards). (3 marks)

Consider temp.
Forward Slice:

```
int temp[size]; // temporary array
    int i = 0;
    while (i < size) {
        if (i=1)
        {
           continue
        }
        temp[size - 1 - i] = arr[i];
        i++;
    }
```

Backward Slice:

```
    int temp[size]; // temporary array
    int i = 0;
    while (i < size) {
       arr[i] = temp[i];
       i++;}
```

d) Perform Information Flow Analysis. (2 marks)

Flow of information
Assignment of values between variables.