MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

(A constituent unit of MAHE, Manipal)

VII SEMESTER B.TECH. (COMPUTER SCIENCE AND ENGINEERING)

MAKE-UP EXAMINATIONS, DEC/JAN 2018

SUBJECT: SOFTWARE TESTING AND ANALYSIS [CSE 4020]

REVISED CREDIT SYSTEM
(29/12/2018)

Time: 3 Hours MAX. MARKS: 50

Instructions to Candidates:

- Answer **ALL** questions.
- Missing data may be suitable assumed.

1A. Differentiate between verification and validation with the help of an example. Write a short note on Y2K software failure. **3M**

1B. A riffle salesperson in the former Arizona territory sold riffle locks, stocks and barrels made by a gunsmith in Missouri. Locks cost $45, Stocks cost $30 and barrels cost $25. The sales person had to sell at least one complete riffle per month and production limits were such that the most a salesperson could sell in a month was 70 locks, 80 stocks and 90 barrels. After each town visit, the salesperson sent a telegram to the Missouri gunsmith with the number of locks, stocks and barrels sold in that town. The system uses locks=-1 to indicate end of input data. When the sales of the month were complete, the Gunsmith computes the salespersons commission as follows: 10% if sales are less than $1000, 15% if sales are between $1000 and $1800 and 20% if the sales are above $1800. The commission program produced a monthly sales report that gave the total number of locks, stocks and barrels sold, the salespersons total dollar sales and finally the commission. **4M**

For the above scenario,

a. Construct Normal Boundary Value Analysis test cases.

b. Identify the valid and invalid classes of the input variables.

c. Write the set of weak normal and weak robust test cases.

1C. Briefly explain the different types of black box testing**.** **3M**

2A. Explain the steps in constructing the decision table. For the scenario given in Q1B. , Construct a decision table and derive the test cases. **4M**

2B. Explain the different types of Equivalence Class Testing techniques. Clearly show how test cases are designed using each of these techniques. **4M**

2C. With the help of an example, explain the various data flow adequacy criteria. **2M**

3A. List the definition, c-use, p-use and DU pairs of all the variables used in the given program and also derive test cases for testing all DU pairs of the program. **4M**

```
1. Main
2 float A,B,C,D, x1, x2
3 boolean is_complex
4 input(A,B,C)
5 D = B*B -4*A*C
6 if D < 0.0
7 then    is_complex = true
8 else  is_complex=false
9 end if
10 if not is_complex
11 then x1 = (-B + sqrt(D)) / (2.0*A)
12         x2 = (-B - sqrt(D)) / (2.0*A)
13 end if
14 output(x1,x2)
15 end
```

3B. How is stub different from a mock object? Under what circumstances we go for mock object? **2M**

3C. Consider the given program P **4M**

```
1.    begin
2.      int x, z, i
3.      input (x,z);
4.      for(i=1; i<5; i++) {
5.      z=z+x*i
6.      }
7.     output( z );
8.    end
```

P is assumed to be correct and hence the expected output is same as the actual output for P.
Test set T is as follows

TId	x	z	Expected
1	0	1	1
2	0	2	2

Mutation Operator M is as follows
< is changed to <=
+ is changed to – [Note: Do not consider ++]
* is changed to +
For the above given problem,
Use FIRST ORDER MUTANTS to test the adequacy of the above given test set. (Represent the results in TABULAR FORMAT wherever applicable. Enhance the test set if required.

4A. Explain Test Minimization and Test Prioritization with an example for each. **5M**

4B. What are the advantages and drawbacks of mutation testing? **2M**

4C. Define a DU pair and Definition Clear path. Give the definition and usage of the following piece of code. **3M**

```
z = &x
y = z + 1
*z = 25
w = *z + 1
```

5A. Consider the following code, **4M**

```
1.    function Mystery (x, y: integer): integer;
2.     var s, z: integer;
3.     s := 1;
4.     z := -1;
5.     if x < 0 then
6.        s := -1;
7.        x := -x;
8.    end if;
9.     if y < 0 then
10.       s := -s;
11.       y := -y;
12.   end if;
13.   while x >= 0 do
14.       x:= x - y;
15.       z:= z + 1;
16.   end while;
17.       z := s * z;
18.    print(z);
19.    return(z);
20.  end Mystery;
```

a. Draw the CFG for the Mystery code.
b. Identify all the independent paths and find the Cyclomatic complexity of the code.
c. Write the test cases for branch coverage, statement coverage and path coverage.

5B. What is the goal of integration testing? When does integration testing occur? **3M**

5C. Explain the different categories of integration errors with the help of a diagram **3M**

****