Reg. No.										

MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

A Constituent Institution of Manipal University

VII SEMESTER B.TECH. (COMPUTER SCIENCE AND ENGINEERING) MAKE-UP EXAMINATIONS, DEC 2018

SUBJECT: SOFTWARE TESTING AND ANALYSIS [CSE 4020]

REVISED CREDIT SYSTEM

(…./…../2018)

Time: 3 Hours MAX. MARKS: 50

Instructions to Candidates:

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

1A.	Differentiate between alpha, beta and acceptance testing with an example.	**4M**
1B.	Consider a problem for the determination of the nature of roots of a quadratic equation. Its input is a triple of positive integers (say a, b and c) and values may be from interval [0, 100]. The output may have one of the following words: Not a quadratic equation, Real roots, Imaginary roots, Equal roots. (Hint: Determinant (D) $=b^2-4ac$. If D=0 roots are equal, if D>0 roots are real.) Design boundary value analysis test cases and robust test cases.	**4M**
1C.	Create efficient set of equivalence classes and generate strong normal test cases for the problem in Q1B.	**2M**
2A.	Create an efficient limited entry decision table for the problem given in Q1B. And also derive the test cases out of it.	**4M**

2B.	For the code shown in Fig Q2B. Write test cases to achieve maximum statement coverage and maximum branch coverage. Compute and analyse the statement coverage and branch coverage metric for your test case.
	
	1 int findSubstring(char str[], char srcstr[])
	
	2 int k=0,j=0,isFound = 0,isMissing=1;
	3 for(int i=0;str[i]!='\0';i++) {
	4 if(str[i]==srcstr[0]) {
	
	5 j=0,k=i;
	6 isFound=1;
	7 for(j=0;srcstr[j]!='\0';j++,k++) {
	8 if(srcstr[j]!=str[k]){
	9 isFound=0;
	10 break;
	11 }
	12 }
	13 if(isFound) {
	14 cout<<"Found at Position:"<<i+1;
	15 isMissing=0;
	16 }
		4M

Reg. No.										

MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

A Constituent Institution of Manipal University

```
17      }
18   }
19   if(isMissing)
20          cout<<"Substring not Found";
21 }
```

Fig Q2B

2C.	When should black box testing and white box testing be used respectively? Why?	**2M**
3A.	For the code in Fig Q2B, draw the CFG (each numbered statement is a node). List all the def-use pairs for the variables:i,j,k,isFound and isMissing. Write test cases to satisfy all-def adequacy criteria.	**4M**

3B.	Consider the program given below, Suppose main () module is not ready for the testing of Largest () module, design and explain how will you test this module in an integration scenario?
	
	<u>Program</u>
	
	`int Largest()`
	`{`
	`  r = 1;`
	`  for (i = 1; i < 3; ++i) {`
	`    if (a[i] > a[r])`
	`        r = i;`
	`  }`
	`  return r;`
	`}`
		2M

3C.	Consider the program given in Fig Q.3C for finding the index of the largest number in an array of three elements. Perform all steps of mutation testing on the test cases given in Fig Q.3.A.2 using following mutant operators:
	Replace 'a[i]' with 'i'
	Replace '>' with '>=' (greater than with greater than and equal)
	Replace ‘a.Length’ with ‘a.Length - 1’
	
	Program
	
	`1  static int Largest(int[] a)`
	`2  {`
	`3    int r = 1;`
	`4    for (int i = 1; i < a.Length; ++i)`
	`5    {`
	`6      if (a[i] > a[r])`
	`7        r = i;`
	`8    }`
		4M

	9 return a[r]; 10 } Fig Q3C Fig Q3C1 Write extra test cases to achieve 100% test adequacy.	
4A.	Explain Test Minimization and Test Prioritization with an example for each.	**5M**
4B.	How is mutation testing different from other testing techniques? Explain the significance of mutation score.	**2M**

4C.	For the given code, list the definition, p-use and c-use of each of the variable used in the code.
	1. int test(int a, int b) {
	2. int k;
	3. int *arr;
	4. k = a;
	5. arr = malloc(sizeof(int));
	6. *arr = k + a;
	7. if (*arr > b)
	8. return (a);
	9. else {
	10. arr = malloc(sizeof(int));
	11. *arr = a + b;
	12. return(*arr);
	13. }
	14. }
		3M

5A.	Consider a 2 variable system, show using diagrams the number of test cases required for: Normal Boundary Value Test Case Robust Boundary Value Analysis Test Case Robust Worst Case (BVA) Test Case Strong Normal Boundary Value Weak Robust Boundary Value Test Case Extrapolate these results to give the formula to find the number of test cases required for an ‘n’ variable system.	**4M**
5B.	What is regression testing? Describe the problems in test selection with a neat diagram. Explain the test all approach with suitable example.	**3M**
5C.	Describe the factors affecting the choice of integration strategy with an example for each. **********	**3M**

	a[0]	a[1]	a[2]
TD1	1	3	1
TD2	1	2	1