EECS3311 Final Exam Winter 2022

Student ID: **Student Name:**

Signature:

Department of Electrical Engineering and Computer Science
EECS3311 – Software Design
Instructor: Song Wang and Ilir Dema
Examination Date and Time: April 26th, 2022, 14:00 PM - 17:00 PM
Examination Location: TC AVIVA

Instructions:

- You have **180 minutes** to complete the exam.
- Unauthorized duplication or archival of these questions is not permitted.
- If information appears to be missing from a question, make a reasonable assumption, state your assumption, and proceed. Do not simplify the question.
- You must answer questions in a clear piece of paper and send your solution to your instructor.
- **Illegible answers receive NO point.**

Question	Mark	Points
Q1 (MCQ)		10
Q2		10
Q3		20
Q4		20
Q5		20
Q5		20
Total:		

Question 1: Multiple Choice Questions (10 points)

1) Suppose you are testing a class Something that includes a public method alpha and a private method beta. Suppose that alpha calls beta. Without changing anything in the class, you can only test beta indirectly. If you could change the access modifiers of one or both methods, what would you recommend in order to make it possible to test beta directly? Circle one.

a. Make beta public.

b. Make beta public during testing and change it back to private afterwards.

c. Use the keyword protected, so that beta will be package private.

d. Use no access keyword, so that beta will be package private.

2) Suppose we have defined this class:

```java
class OddballException extends RuntimeException { . . . }
```

Suppose we have a method called `someCalculation`, and that it calls a helper method that throws `OddballException`. Which of the following is true? Circle one.

a. `someCalculation` must put that call inside a try-catch clause.

b. `someCalculation` must declare that it may throw a RuntimeException.

c. `someCalculation` must either put that call inside a try-catch clause or declare that it may throw a RuntimeException.

d. Trick question! You can't extend RuntimeException.

e. None of the above.

3) Assume we would like to compare the amount of design for the same project, developed using two different methodologies: Waterfall and Agile (Scrum). Evaluate the statement: The amount of product design carried out in the Waterfall Model is much larger than amount of product design carried out in Scrum Framework.

a. True

b. False

4) Which of the following statements is true:

a. A Java interface can implement other Java interfaces

b. A Java interface can extend other Java interfaces

c. A Java abstract class cannot implement more than one Java interface

d. All Java interfaces inherit methods from Object class

5) Consider the following Java code:

```java
public class C {
  int i;
}
public class D extends C {
  void m() {
    this.i = 3;
}}
```

Evaluate this statement: The given code compiles as long as C and D are not in the same package.

a. True

b. False

6) All the methods in both Abstract and Interface classes must be implemented in the subclasses.

a. True

b. False

7) Consider the following Java code:

```java
public class Address {
// ... some code
}
public class Student {
private Address homeAdress;
// constructor, code ...
// in particular, this class has no setter for homeAddress
public Address getHomeAddress() {
return this.homeAddress;}
}
```

Evaluate this statement: Other classes cannot update the homeAddress property of a Student object because this property is private and no setter has been provided.

a. True

b. False

8) All the design patterns can help reduce the complexity of software projects.

a. True

b. False

9) In DBC, any contract in a program can be considered as the requirements of the program.

a. True

b. False

10) All the methods in both Abstract and Interface classes must be implemented in the subclasses.

a. True

b. False

Question 2: Concept Explanation (10 points)

(a) What is the difference between `Randoop` and `Evosuite` regarding test case generation? [2 points]

(b) What is the difference between Aggregation relationship and Composition relationship between classes? [2 points]

(c) What is the difference between Prototype model and Spiral model in software process modeling? [2 points]

(d) What is the difference between Builder Pattern and Prototype Pattern? [2 points]

(e) What is the difference between Adapter Pattern and Bridge Pattern? [2 points]

Question 3: Test Case Management (20 points)

Test case prioritization (TCP) schedules test cases for regression testing in an order that attempts to maximize specific objectives, e.g., achieving code coverage at the fastest rate possible. TCP is widely used in real industrial practices, usually, code coverages of test cases come from the last execution of the regression test suite. `Total statement coverage` and `additional statement coverage` are two kinds of coverage-based test case prioritization strategy. Their definitions are given as following:

`Total statement coverage` : sorts test cases in descending order of the total number of statements covered by each test case. If two test cases cover the same number of statements, randomly select one.

`Additional statement coverage` : selects, in turn, the next test case that covers the maximum number number of statements not yet covered. If two test cases cover the same number of statements, randomly select one.

In this question, you will try these two kinds of test case prioritization strategies on a sample program list in the following table 1.

Table 1: Dynamic code coverage of the test cases.

Statements/Tests	**TC1**	**TC2**	**TC3**	**TC4**	**TC5**	**TC6**	**TC7**	**TC8**
s1		x	x					
s2	x	x			x			
s3			x			x		
s4	x						x	
s5			x			x		
s6							x	x
s7	x		x			x		
s8		x		x				
s9				x		x		
s10		x	x		x			
s11					x			x
s12		x		x	x			

(a) Please, give test cases prioritization orders with respect to `total statement coverage`. [6 points]

(b) Please, give test cases prioritization orders with respect to `additional statement coverage` (please list at least 20 different orders). [6 points]

(c) In this example, which strategy do you think is better? and Why? [4 points]

(d) Code coverage of test cases collected from the last execution of the regression test suite might be not accurate, do you know why? [4 points]

Question 4: Control Flow Graph (20 points)

Below is a python implementation of exponentiation by squaring: `https://en.wikipedia.org/wiki/Exponentiation_by_squaring`

```
1  def exp_by_squaring_iterative(x, n):
2   z= abs(n)
3   r = 1.0
4   while z > 1
5     if z % 2 == 0:
6       x = x * x
7       z = z / 2
8     else:
9       r = r * x
10      x = x * x
11      z = (z-1)/2
12   r = r * x
13   if n < 0;
14     r = 1.0 / r
15   return r
```

(a) Draw the minimal node (hint: 8 nodes) basic block control flow graph (CFG) for the function `exp_by_squaring_iterative`. You must use only the line numbers provided above as the content of each node, and label the nodes using only circled lowercase letters (i.e., nodes ⓐ, ⓑ, ⓒ, ⓓ, ⓔ, ⓕ, ⓖ, and ⓗ). Ensure the graph is clear and legible. Unclear graphs will not receive full points. [15 points]

(b) What's the branch coverage for input x =1; n=2? Please show your calculation process. [5 points]

Question 5: Design By Contract (20 points)

Suppose we implement the HeapSort algorithm with the following code. HeapSort divides its input into a sorted and an unsorted region, and it iteratively shrinks the unsorted region by extracting the largest element from it and inserting it into the sorted region. Heap sort maintains the unsorted region in a heap data structure to more quickly find the largest element in each step.

The HeapSort algorithm can be divided into two parts. In the first step, a max heap (root node is always great or equal to left/right nodes) is built out of the data (see line 27). The heap is often placed in an array with the layout of a complete binary tree. The complete binary tree maps the binary tree structure into the array indices; each array index represents a node; the index of the node's parent, left child branch, or right child branch are simple expressions. For a zero-based array, the root node is stored at index 0; In the second step (starts from line 32), a sorted array is created by repeatedly removing the largest element from the heap (the root of the heap), and inserting it into the array. The heap is updated after each removal to maintain the heap property. Once all objects have been removed from the heap, the result is a sorted array.

```
 1  public class HeapSort {
 2  // Given a node at index i, arr[(i-1)/2] Returns the root/parent node.
 3  // arr[(2*i)+1] Returns the left child node. arr[(2*i)+2] Returns the right child node.
 4  public static void heapify(int [] arr, final int i, final int len) {
 5    int j = i;
 6    while (true) {
 7      int m = j;
 8      if (m <= len/2) {
 9        int c = 2*m;
10        if (arr[c-1] < arr[m-1]) m=c;
11        if (c < len && arr[c] < arr[m-1]) m=c+1;
12      }
13
14      if (m==j) break;
15      int tmp = arr[j-1];
16      arr[j-1] = arr[m-1];
17      arr[m-1] = tmp;
18      j = m;
19    }
20  }
21
22
23  public static void sort(int [] arr) {
24    // Array of size 1 is already sorted
25    if (arr.length < 2) return;
26
27    // Step 1: Build heap
28    for (int i = arr.length/2; i > 0; i--) {
29      heapify(arr,i,arr.length);
30    }
31
32    // Step 2: Now sort
33    int tmp;
34    for (int len = arr.length-1; len>1; len--) {
35      tmp = arr[0];
36      arr[0] = arr[len];
37      arr[len] = tmp;
38      heapify(arr,1,len);
39    }
40    tmp = arr[0];
41    arr[0] = arr[1];
42    arr[1] = tmp;
43  }
44  }
```

Please finish the following contracts for class `HeapSort`:

(a) Pre-condition: the first input, i.e., "`int [] arr`", of `heapify` should not be null or empty. [2 points]

(b) Pre-condition: the inputs, i.e., "`i`" and "`len`" should be less than or equal to the size of "`int [] arr`". [2 points]

(c) Loop Invariant of `while` statement (line 6): during the process of heap building, each root is great than or equal to its left child node. [4 points]

(d) Loop Invariant of `while` statement (line 6): during the process of heap building, each root is great than or equal to its right child node. [4 points]

(e) Post-condition: after executing `sort(int [] arr)`, the elements in `arr` are sorted in descending order. [4 points]

(f) Post-condition: the elements in `arr` are the same before and after executing `sort(int [] arr)`. [4 points]

Question 6: Design Pattern Practice (20 points)

Visitor design pattern is a behavioral design pattern that lets you separate algorithms from the objects on which they operate. In Visitor pattern, we use a visitor class which changes the executing algorithm of an element class. By this way, execution algorithm of element can vary as and when visitor varies. As per the pattern, element object has to accept the visitor object so that visitor object handles the operation on the element object.

Given the following example project that use visitor design pattern:

We are going to create a `ComputerComponent` interface defining *accept* operation. It has four concrete subclasses, i.e., `Keyboard`, `Mouse`, `Monitor`, and `Computer` that implement the `ComputerComponent` interface. There is another interface `ComputerComponentVisitor` that defines a *visitor* operation, we implement the `ComputerComponentVisitor` interface with a concrete class `ComputerDisplayVisitor`. `VisitorDemo` is demo and entrance class of this project, it use `Computer` and `ComputerDisplayVisitor` classes to demonstrate use of visitor pattern.

In this question, you task is to draw a UML class diagram to reflect the above project designed with Visitor pattern.