CSE3311 Software Design
Summer 2013
Midterm Exam

Name: ______________________

Student#: ______________________

CSE Account: ______________________

Points: ____________ **/100**

Instructions:

1. This is a closed book examination. No examination aids are permitted no calculators, telephones, etc.
2. All questions are to be attempted.
3. Answer each question in the space provided.
4. Each question value is provided in brackets next to the question number.
5. Where descriptive answers are requested, use complete sentences and paragraphs
6. All programming and mathematical work is to be annotated; include comments explaining what you are doing.
7. Wherever appropriate, use diagrams to enhance your answers.
8. If a question is ambiguous or unclear then please write your assumptions and proceed to answer the question.
9. You may write in pencil.
10. Do not use a red ink
11. The examination time is approximately three hours (180min).
12. **Write legibly. Illegible writing that cannot be deciphered will not be marked!**
13. This exam contains 17 pages. Please check the completeness of your booklet.

1. (_______/*10 points*) Multiple choice questions

1. What is a role of a *class invariant*?

   A. It is used to validate the correctness of the feature's execution
   B. It is used for a constant validation of the object's state
   C. It is used to validate the correctness of parameters' values
   D. It is used to validate the loop execution

2. What is a role of a *require statement*?

   A. It is used to validate the correctness of the feature's execution
   B. It is used for a constant validation of the object's state
   C. It is used to validate the correctness of parameters' values
   D. It is used to validate the loop execution

3. Class is combination of two concepts from OO context. Which are these concepts?

   A. Object
   B. Module
   C. Type
   D. Client

4. What does it mean that *modules should be both Open and Closed* ?

   A. modules should be open for modification by the owner but closed for modification by other developers
   B. modules should be encrypted (closed) and could be decrypted (opened) with a key
   C. modules should be open for modification and have stable API
   D. modules should have open specification and closed implementation

5. Inheritance from the **module** point of view is?

   A. if B inherits from A, all the services of A are potentially available in B
   B. if B inherits from A, all the services of B are potentially available in A
   C. If B inherits from A, whenever an instance of A is required, an instance of B will be acceptable
   D. If B inherits from A, whenever an instance of B is required, an instance of A will be acceptable

6. Select the **strongest** assertion from those provided below.

   A. $True$
   B. $y = a * a \wedge a > 0 \wedge a < 100$
   C. $y = a^2 \wedge a >= 0 \wedge a < 5$
   D. $False$

7. From the **module** viewpoint: if B inherits from A ...

   A. all the services of B are potentially available in A
   B. all the services of A are potentially available in B
   C. whenever an instance of B is required, an instance of A will be acceptable
   D. whenever an instance of A is required, an instance of B will be acceptable

8. Ancestors of class B are ...

   A. class B and all classes that inherit from B
   B. the parents of B and their ancestors
   C. only the parents of B
   D. B itself and the ancestors of its parents

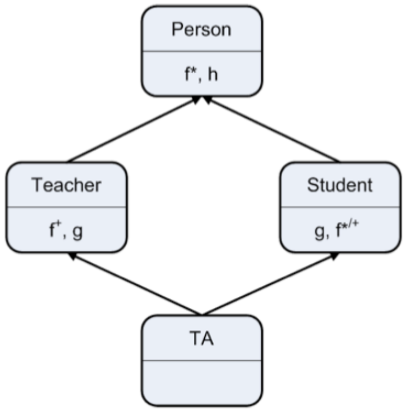
9. *Whenever a software system must support a set of alternatives (e.g. variants of a figure in graphic systems), one and only one module in the system should know their exhaustive list*
   The statement above is a part of which OO principle?

   A. Single Choice Principle
   B. Open-Closed Principle
   C. Liskov Substitution Principle
   D. Single Responsibility Principle

10. What happens with pre- and post-conditions of the feature when the feature is inherited and redefined?

    A. only new pre- and post-conditions are used (old are ignored)
    B. precondition and postconditions are connected by OR
    C. precondition is connected by OR and postconditions by AND
    D. precondition is connected by AND and postconditions by OR

2. (_______/*10 points*) Consider the following inheritance diagram. What problems can arise and how can they be resolved? Consider what happens when function f in class Student is deferred (*) or effective (+). You are not required to solve all problems simultaneously. Describe each problem independently and propose the way(s) it can be resolved.



Answer:

- *g* is defined in both Teacher and Student (*2poits*). This can be solved by renaming of both methods (*1 poit*)
- if *f* is implemented by the Student then we have a problem with a conflict (*2poits*) that can be saved by renaming features (*1 poit*)
- if *f* is differed in Student then there is no problem (*1 poit*)
- *h* is inherited in two paths however not modified in neither path, there is no problem(*1 poit*)

You can rename, or undefine and select to resolve the problem (*2 poits*).

3. (_______/*10 points*) In mathematical notation, give the indicated contract assertions that are provided in comments below. **Do not use agents.**

Business rules:
Each study group has a name and a list of its members. Each student has a name and a list of the names (ids) of the study groups in which they want to be members.

```
class STUDENT
   name: STRING
   in_group: SET[STRING]
end
```

```
class STUDY_GROUP
   name: STRING
   members: SET[STUDENT]
end
```

```
class STUDENT_ASSOCIATION
   study_groups : SET[STUDY_GROUP] -- All the study groups in the association
   members : SET[STUDENT]   -- All the students in the association



   make(students : SET[STUDENT] , group_names: SET[STRING])
      -- Creates an association with the study groups according to the preferences of the students.
      -- students  the students making the association
      -- group_names  the list of names of all the study groups in the association



      require
      -- The names of the study groups that students want to be in are in group_names.
```

Answer:

$$\forall s \in students \cdot s.in\_group \subset group\_names$$

alternate

$$\forall s \in students \cdot (\forall sg \in s.in\_group \cdot sg \in group_names)$$

```
      -- Between 3 and 5 students can to be in each study group.
```

Answer: $\forall name \in group\_names \cdot 3 \leq \#s \in students | name \in s.in\_group \cdot s \leq 5$

```
   ensure
      -- The names of the study groups that students want to be in are in group_names.
      -- Between 3 and 5 students can to be in each study group.
      -- There are no members of the association other than the students making
      --   the association and all
      -- the students are members of the association.
```

Answer: First two are in require.

$$members = students \ (0.5)$$

alternate

$$\forall m \in members; s \in students \cdot m \in students \land s \in members$$

```
      -- The study groups in the association are precisely those with names in group_names.
```

$$group\_names = \{sg \in study\_groups \cdot sg.name\} \ (0.5)$$

alternate

$$(\forall sg \in study\_groups \cdot sg.name \in group\_names) \land \#study\_groups = \#group\_names$$

```
   number_of_groups(group_size : INTEGER) : INTEGER
   -- Returns the number of study groups that have group_size members

      ensure
```

$$\boxed{Result = \#\{sg \in study\_groups | \#sg.members = group\_size \cdot sg\}}$$

```
   invariant
   -- The members of every study group are all the people who want to be in that study group.
```

Answer:
$\forall sg \in study\_groups \cdot sg.members = \{m \in members | sg.name \in m.in\_group\}$
alternate
$\forall sg \in study\_group; m \in members \cdot m \in sg.members \iff sg.name \in m.in\_group$

```
end  STUDENT_ASSOCIATION
```

Each statement worth 2 points except for the two marked as 1 point. 10 in total

4. (_______/*10 points*) We have a group of related objects such as the following that need to be shared among several classes, say A,B, C. Describe, in the context of object-oriented design, two different methods of sharing these global objects.

```
Pi:  REAL is 3.141592
Mean_radius_of_earth:  REAL is 6371.01 -- mean radius of earth in kilometers
```

Method 1: Description(*3 points*):

Inheritance from CONSTANTS – classes that require constants inherit from the class that defines them.

What is the major advantage of using this method?(*1 point*)

Have direct access to the variables, no need to introduce other variables

What is the major disadvantage of using this method?(*1 point*)

It is incorrect from semantic point of view because the call that "uses" constants not necessarily "is" constants

Method 2: Description(*3 points*):

Singleton – a pattern that assures that there is a single instance of the class

What is the major advantage of using this method?(*1 point*)

Single copy assures consistency

What is the major disadvantage of using this method?(*1 point*)

Adds complexity

5. (_______/*10 points*)

A) Consider the following program text.

```
class THING feature
   thing: THING is once
      Result := Current
   end
   invariant
      only_thing: thing = Current
end

class BETTER
   inherit THING
end

class WORSE
   inherit THING
end
```

Assuming the class invariant is checked, does the execution of the following statements lead to any assertion violation in Eiffel? **Justify your answer.**

```
      thing_1, thing_2 : THING
      the_worse: WORSE
      the_better: BETTER
      create the_worse
      create the_better
      thing_1 := the_better.thing
      thing_2 := the_worse.thing
```

Answer: *Yes! When "create the_better" is executed, the class invariant in THING will become false (violated). Current in THING refers to the_better whereas thing points to the_worse.*

B) Assume class `BETTER`, from part A, is modified as defined below. Draw memory diagrams showing objects created as a result of execution of the set of statements in Part A. Assume assertions are turned on.

```
class BETTER
   inherit THING
      redefine thing end
   feature
      thing: BETTER
         once Result := Current
      end
end
```

thing_1 has the same value as the_better and thing_2 has the same value as the_worse.

6. (_______*/10 points*) Answer the following questions:

A) Does the loop invariant depend on the postcondition (ensure)? You must justify your answer.

Answer: *Yes! Since it should be used to prove the correctness of the loop. If you change the postcondition, then the loop invariant needs to be changed, and vice-versa, as there is a co-dependency.*

B) Does the loop invariant depend on the loop implementation? You must justify your answer.

Answer *Yes! This is an assertion about the loop property. It is about the variables in the loop and their relationships. If you change the implementation, then the loop invariant needs to be changed, and vice-versa, as there is a co-dependency.*

7. (_______*/10 points*)
Using the given assertions, prove the following algorithm works correctly. The more mathematically precise you are and the better the annotation, the higher the grade because it is less ambiguous.If you find an error in the code, apply the fix based on your proof and finish the proof.

`Pre-condition:` $a : INTEGER \wedge b \geq 0$
`Post-condition:` $z = abs(a + 2 * b)$ `-- abs is the absolute value, z > 0`
`Loop invariant:` $z = a + 2 * (b - y)$

```
1    z := a
2    y := b
3    while y /= 0 //(was =)
4       z:=z+2
5       y:=y-1
6    end
7    if z < 0 then z := z
```

Proof

Question 0: loop invariant is given
Question 1 – base case – Is loop invariant true after initialization phase, before loop body Loop initialization occurs in lines 1 and 2 giving the relationships: $z = a \wedge y = b$
Substitute the values of z and y into the loop invariant to get the following expression. $a = a + 2 * (b - b)$ and simplifying gives a = a which is true. The base case is established

Question 2 inductive case: Assuming LI is true before executing loop body, and loop condition permits executing the body, is LI true after executing the loop body.

$$Canworkforwardsorbackwards; hereweworkforwards.$$

Executing the body gives the relationships: $z' = z + 2 \wedge y' = y - 1$

Solve for z and u, and substitute into the loop invariant
$z = a + 2 * (b - y)$ //the loopinvariant
$\rightarrow z' - 2 = a + 2 * (b - (y' + 1))$ //substitute for z and y
$\rightarrow z' = a + 2 * (b - y' - 1) + 2$ //Add 2 to each side and do inner subtraction
$\rightarrow z' = a + 2 * (b - y') - 2 + 2$ //Take the -1 out of the ()
$\rightarrow z' = a + 2 * (b - y')$ //the 2's cancel

The final expression is the loop invariant at the end of the loop body. Since all we did was substitute equals for equals and rearrange terms using valid arithmetic operations, and the original LI is true, then it follows that the LI at the end of the loop body is true.

Question 3a: Does the loop terminate?
$y \geq 0$ at line 3 because $b \geq 0$ from the precondition and $y = b$ from line2.
y is decremented by 1 every time around the loop, therefore eventually y must be equal to zero and the exit condition $y = 0$ must become true.

Question 3b: Does exit condition and LI imply post-condition? At loop termination the loop invariant is true. The following substitutions and rearrangements are made

$z = a + 2 * (b - y)$ //the loopinvariant
$\rightarrow z = a + 2 * (b - 0)$ //Substitute y = 0
$\rightarrow z = a + 2 * b$ //(1)

Now if at line 7 we have two cases.
Case 1: $z < 0$ and the then-phrase executes with the relationship $z' = -z$ As a consequence relationship (1) is equivalent to $z = abs(a + 2 * b)$

Case 2: $z \geq 0$ and relationship (1) is equivalent to $z = abs(a + 2 * b)$ In both case the relationship is $z = abs(a + 2 * b)$ true. But this is exactly the post condition of the algorithm. As a consequence the algorithm is correct with respect to the given conditions

8. (_______/*15 points*)
For the following requirements identify candidate classes (with appropriate names) and define their relationships using BON. Use compressed notation to represent classes. Make sure to show all association and aggregation labels. Explain why you chose the relationships (is_a and has_a) in your diagram. If any of the requirements should be implemented as class invariants, indicate the class that must contain the invariant, explain why, and write the invariant as a mathematical expression.
Requirements:

- A coffee shop offers several types of hot drinks including coffee, hot chocolate, and tea.
- The price of tea is less than the price of coffee and coffee is cheaper than hot chocolate.
- Price of the drink can be modified on run-time.
- Shop has a sitting area with a minimum of 10 tables.
- A coffee shop has at least one owner.
- Each drink can have additions such as extra white or brown sugar, honey, regular, skim or 0% milk, half-and-half, soy milk, and variety of flavour shots.
- Some extras can increase the price while others may be free (e.g., favour shots are \$0.5 while sugar is free).

Design:

9. (_______*/15 points*)

A) We learned that software modelling is difficult: by changing our viewpoint we can rephrase an "is-a" relationship to a "has-a" relationship. For example "every software engineer is an engineer" can be rephrased as "every software engineer has an engineer component", therefore changing "inheritance" relationship to "client-supplier". Discuss what criteria can be used, in such situations, to decide whether is-a or has-a is the most appropriate relation for the chosen entities.

See OOSC 24.2, page 812 for discussion.

B) We have two relations between classes: client-supplier and inheritance. Compare these two relations in terms of reuse, information hiding and protection against change. That is to say which one of these relationships allow for reuse of services provided by a class, which allows for information hiding, and which protects a class against change. Justify your answers.

OOSC page 609 for discussion

C) In Eiffel a descendent can redefine a function with no argument into an attribute, but not vice versa. First explain why it is not possible to redefine an attribute into a function. Then explain how a descendent defining a function into an attribute should handle the postcondition of that function.

OOSC pages 491-493

(Notes)

Answers

1. B 2. C 3. B 3. C 4. C 5. A 6. D 7. B 8. D 9. A 10. C