

**THE AUSTRALIAN NATIONAL UNIVERSITY**

*First Semester 2009 Semester Final Examination*

**COMP2100/2500**  
**Software Construction**  
**&**  
**Software Construction for Software Engineers**

*Writing Period: 2 hours*

*Study Period: 15 minutes*

*Permitted Materials: One page A4 notes both sides.*

*Answer all questions*

*Write your answers in the boxes provided in this booklet. Only those answers written in this booklet will be marked. There is additional space at the end of the booklet in case the boxes provided are insufficient. Label any answers you write at the end of the booklet to indicate which question they refer to. Do not remove this booklet from the examination room. Do not write in red ink anywhere in the booklet.*

*Marks add up to a total of 100. Questions have unequal values.*

Write your student number here:

Student Number:

Official use only:

Q1 (18)	Q2 (15)	Q3 (25)	Q4 (22)	Q5 (20)	Total (100)

## QUESTION 1 [18 marks]

- (a) Define and distinguish between *boundary value* testing and *equivalence class* testing.

QUESTION 1(a)	[3 marks]

- (b) Define and distinguish (state what is the important difference between them) the terms *path testing* and *boundary value testing*.

QUESTION 1(b)	[3 marks]

- (c) In the Subversion version control system (SVN), what are the four states that a working copy file can have, in relation to the repository.

QUESTION 1(c)	[3 marks]

(d) Consider this fragment of Java code

```
public abstract class Tree {
    abstract public void Traverse();
}
public class NullTree extends Tree {
    public NullTree() { /* do nothing */}
    public void Traverse() { /* do nothing */ }
}
public class BiTree extends Tree {
    private int value; private Tree left, right;

    public void Traverse() {
        left.Traverse();
        right.Traverse();
        System.out.println(value);
    }
}
```

Which kind of binary tree traversal is implemented by this code?

Circle one letter to indicate your answer.

[2 marks]

- A. inorder
- B. preorder
- C. postorder
- D. none of the above

(e) The characters ‘;’ (semicolon) and ‘|’ (vertical bar) have special meanings in the syntax of bash shell commands. State the meanings of these two characters in bash.

QUESTION 1(e)	[4 marks]

(f) What are the essential features that distinguish an object oriented (OO) language from a non-OO, procedural language?

QUESTION 1(f)	[3 marks]



- (c) Compared to working with two programmers, how does having a third programmer in the team change the way each programmer needs to work and interact with the version control system? Explain your answer to with a clear example of a file that is being modified by editing by all three programmers in several sessions, and considering what states each of the programmers' working copies and the repository may be in.

QUESTION 2(c)

[7 marks]

### QUESTION 3 [25 marks]

Consider the design of tests for a program with the following specification:

One component of a phone call billing application is a program named *CappedCharge*. This program reads lines of data each of which lists a set of phonecalls for one phone account. Each call is one of three different call-types, identified as types A, B and C.

The program contains a preset built-in table with two entries for each call-type: (1) the charging rate (in whole cents per second) and (2) the “charge cap” (a whole number of cents) or an entry showing ‘no cap’ if there is no cap on that call-type.

For example:

*Assume that this charge table is fixed data and is not varied in these tests.*

call type	rate	cap
A	10	1500
B	35	540
C	40	no cap

The program reads a line of data which contains:

- a total charge cap on the account (in whole cents) and
- a list of call data which is in pairs: a call-type (denoted by a letter) and the duration of the call (in seconds).

For example, one input might be `2000 A 10 B 100 C 60 A 15 A 10`

#### Functional specifications

1. The *raw charge* for a call is the duration of the call multiplied by the rate for that type of call
2. the *capped charge* for a call is calculated as the smaller of (1) the raw charge for the call and (2) the charge cap for that type of call, if there is any cap;
3. the *raw total charge* is calculated as the sum of the capped charges for all calls
4. the *capped total charge* is calculated as the smaller of (1) the raw total charge and (2) the cap for the account
5. the CappedCharge program shall print out the capped total charge

**Do NOT design the program.** Answer the questions about choosing test cases below.

Write your answers to these questions in the format shown here: for each test case give both a test input and the expected output.

For example, one test case could be written:

Input: 2000 A 10 B 100 C 60 A 15    Expected Output: 3190

This input means: an account cap of 2000 (cents); with 4 calls: one call of type A, duration 10 seconds; one call of type B, 100 seconds; and one call of type C, 60 seconds duration; and another call of type A, 15 seconds. The expected output is stated to be 3190.

The test case designer decides to do equivalence testing. She partitions the behaviour space for this program in the following way, and intends to design equivalence class test cases for each meaningful combination of the input values that define these classes:

1. the raw total charge is (a) above or (b) below the account cap
2. the number of calls in the call list is (a) zero (b) one (c) more than one
3. the number of call-types in the input list of calls: *the designer has not decided*
4. the number of calls of one call-type: (a) zero (b) one (c) more than one
5. the raw charge for one call is (a) below (b) above the cap for that call-type; or (c) for a call-type with no cap

(a) Design one simple test case that tests case 2(a): zero calls in an input list. (give only one test case: include both the Input and Expected output)

QUESTION 3(a)	<b>[3 marks]</b>

(b) Design one test case as simple as possible for the combination of 1(a) and 2(c) and 5(b)

QUESTION 3(b)	<b>[4 marks]</b>

(c) How should the designer partition the program behaviour space for partition 3 (the number of call classes in one dataset)? Briefly explain your answer.

QUESTION 3(c)	<b>[5 marks]</b>

*question continues on the next page*

- (d) Using the principle of Boundary Value testing, write one or more test cases to fully test the boundaries implied by Functional Specification 2: the capped charge for a call. Choose data for call type A in your test cases.

QUESTION 3(d)	[6 marks]

- (e) The equivalence class partitions listed above do not consider several aspects of the call type data: for example, what order they come in (all of one call type before all of any other type, or interleaved at random); whether there is more than one of each type of call in one dataset; whether the calls come in increasing length of time, or the order is not constrained, and so on.

Discuss whether these factors should be included in the testing of this program. Use supporting arguments from the principles of black box test case design and the nature of the equivalence classes for this specification.

QUESTION 3(e)	[7 marks]

*answer box continues on the next page*











\* this page is blank - replace with 2 column program listing protoWidget page 1

\* this page is blank - replace with 2 column program listing protoWidget page 2

## QUESTION 5 [20 marks]

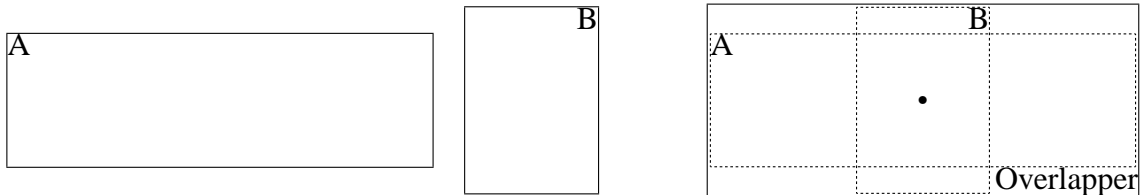
Consider a prototype program that describes containers in a graphics display system. It contains two kinds of container elements which have either a vertical layout or a horizontal layout; and two types of graphic widget elements, namely a text block and a coloured “filler” rectangle. The only aspects that have been modelled so far are the width and height of elements. See the program listing on the previous page.

You do not need to understand any details of graphics for this question: these Widgets and Containers behave in some ways like the Oops program `XmlElement` and `XmlContainerElement`, with visitors like those that compute the value in Expression trees.

We define a new kind of Widget which contains exactly 2 children elements.

```
public class Overlapper extends Widget {  
    Widget childA, childB; ...}
```

The graphical layout for an `Overlapper` Widget overlaps these elements at a common centre. The combined width of the `Overlapper`'s bounding box is therefore is the maximum of the widths of its two elements, and the combined height is the maximum of their heights.



(a) Write the `BoundingBox` visitor method `visitOverlapper`.

QUESTION 5(a)

[10 marks]





Additional answers to QUESTION —(—)[—]

Additional answers to QUESTION —(—)[—]