

**THE AUSTRALIAN NATIONAL UNIVERSITY**

*First Semester 2007 — Midsemester Examination*

**COMP2100/2500/6244  
Software Construction**

*Writing Period: 60 minutes*

*Study Period: 10 minutes*

*Permitted Materials: One A4 page with notes on 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 60. Each Part has equal value; questions have unequal values.*

Name (family name first):

Student Number:

*Official use only:*

Q1 (5)	Q2 (10)	Q3 (15)	Q4 (15)	Q5 (15)	Total (60)

**Part 1. Testing. 15 marks**

**QUESTION 1 [5 marks]**

(a) Briefly describe what is a *boundary* test case

QUESTION 1(a)	[1 mark]

(b) Briefly describe what is an *equivalence class* test case.

QUESTION 1(b)	[1 mark]

(c) Briefly describe the difference between these two kinds of test cases.

QUESTION 1(c)	[1 mark]

(d) State what are the most important reasons that make boundary test cases so *effective* in discovering defects in a program.

QUESTION 1(d)	[2 marks]

## QUESTION 2 [10 marks]

Consider designing tests for a program with the following specification:

The ShortNumToWords program shall take exactly one command line argument.  
The argument should be a positive integer less than one thousand

- 1.1 If the program gets the wrong number of arguments, *or*
- 1.2 if its argument isn't an integer, *or*
- 1.3 if the argument is not in the range 1-999, *then*  
it shall print an appropriate warning message and exit.
- 2 If the input is OK it shall print out the value of its argument written out in words.

These specifications are taken from homework 6 but you do not need to have completed this homework to answer the question. **Do NOT design the program**, answer the questions below.

State answers to these questions in the form of *test cases*: for each test case give both the test value *and* the expected output).

For example, one test case could be written:

*Input:* ShortNumToWords 66    *Output:* sixty-six

- (a) what is one of the **boundary value** test cases for requirement 1.1 (wrong number of arguments)? (*choose just one test case*)

QUESTION 2(a)

[1 mark]

- (b) what are 2 **boundary value** test cases for requirement 1.3 (argument out of range)?

QUESTION 2(b)

[3 marks]

- (c) what are 2 test cases in different **equivalence classes** for requirement 1.3 (out of range)?

QUESTION 2(c)

[2 marks]

- (d) what are 2 test cases in different **equivalence classes** for requirement 2 (OK input)?

*There may be more than 2 equivalence classes for this requirement—choose only 2 cases.*

QUESTION 2(d)

[4 marks]

**Part 2. Version Control. 15 marks.**

**QUESTION 3 [15 marks]**

- (a) Some version control systems use *locking* and some are *non-locking* systems. What kind of system is Subversion (SVN)?  
(Note: a wrong answer here will be given a *negative* mark.)

QUESTION 3(a)	[2 marks]

- (b) What are the advantages and disadvantages of a non-locking version control system?

QUESTION 3(b)	[4 marks]

- (c) Give an illustrative example of how Subversion (SVN) can be used effectively by 2 people (call them A and B) who are working collaboratively on a project. They sometimes need to change different parts of the one file at almost the same time and then continue working with the most up-to-date version. Assume that the base version of the files are already in the repository, and both people already have made working copies. Indicate the actions of A and B and possible version numbers of the files, to make your example as clear as possible.

QUESTION 3(c)	[9 marks]



### Part 3. 15 marks. Recursive Data Structures

**QUESTION 4 [15 marks]** This question is based on the example of modelling simple expressions, presented in lectures as `Expression4`. Line numbering labels (A1..A10, E1..E11 etc.) have been added to mark some lines in parts of the code listed on the next page.

- (a) When the program is executed, evaluating the expression `c.accept(f)` at line D8 in `demonstrateExpr` will cause a sequence of method calls which will assign a series of new values for the attribute `f.string`.

Show the sequence of `visit...` and `accept` method calls;

at each step in this sequence, state the line number of the statement being executed in the `ExpressionFormatter` object `f`, or in any `Constant4` or `Addition4` object.

Include which of the objects (named `x`, `y`, `c`, `d`, `f`) is involved at each step, and any changes to the value of `f.string`.

The first three steps are shown below:

Initially `f.string` has the value `"` (empty string)

1. `c.accept(f)` line A11: call `f.visitAddition(c)`
2. `f.visitAddition(c)` line E6: `f.string` is `"`
3. `f.visitAddition(c)` line E7: call `x.accept(f)`

You can stop as soon as `f.string` has the `+` included.

QUESTION 4(a)	[9 marks]

- (b) We want to add a second kind of constant for real numbers (`Real4`), and to change the existing classes to treat integers and reals in equal ways.

What changes must be made to the data structures and methods in the program?

A real number value can be represented by the type **float** in Java.

Write out any new methods and structures in the space below the program listing and on the Additional Answers pages, and clearly mark any changes that must be made to the existing program code. Do not write anything else on the program listing.

QUESTION 4(b)	[6 marks]
---------------	-----------

\* this page is blank - replace with 2 column program listing Expression4

## Part 4. 15 marks. Programming based on Homework.

**QUESTION 5 [15 marks]** Homework 5 is the LineCount program: counting logical lines of code in multiple Java program files. The original specification is

The program **LineCount** accepts any number of command line arguments, each assumed to be the name of a Java source file. For each of those files it must open the file and count the number of lines of code. It must count all lines *except* those that are empty, or contain only:

- whitespace characters (spaces and tabs)
- left or right braces: `}` or `{`
- Comment start, finish or continuation markers: `/\*`, `/\*\*`, `\*/`, `\*/` or `//` or any combination of those.

After scanning each file the program must write a report line to the standard output stating the number of lines of code in that file, and the file's name. After it has done that for each file named in the command line arguments, if there was more than one file then it must leave a blank line and then print a summary line giving the number of files read and the total number of lines of code.

A program to do this is on the next page. Note that it contains a method called WordCount that is not currently used.

Your task is to design changes to this program to meet the following specification:

The program **WordLineCount** accepts any number of command line arguments, each assumed to be the name of a Java source file. For each of those files it must open the file and count the number of lines **of any kind. All lines are counted, none is ignored.**

**It also counts the number of "words" in the file. A word is defined to be a token as determined by the StringTokenizer method nextToken.** (*This has been implemented for you, in the WordCount method.*)

After scanning each file the program must write a report line to the standard output stating the number of lines of code **and the number of words** in that file, and the file's name. After it has done that for each file named in the command line arguments, if there was more than one file then it must leave a blank line and then print a summary line giving the number of files read and the total number of lines of code **and total number of words.**

Show your answer neatly **on the program listing**, showing clearly which lines you want deleted, which lines are changed, and what lines are inserted and exactly where.

Note: this task is possible with less than 15 lines of changes.

QUESTION 5

[15 marks]

\* this page is blank - replace by two column landscape program listing LineCount.java

