

THE AUSTRALIAN NATIONAL UNIVERSITY

First Semester 2006

Special and Supplementary Exam

**COMP2100/2500
Software Construction**

Writing Period: 3 hours 30 minutes

Study Period: 15 minutes

Permitted Materials: The comp2100/2500 lab notebook

Instructions

1. Answer all four questions.
2. The questions have equal value.
3. Marks for each part are indicated on the exam paper.
4. Answer all questions by creating or modifying files on the computer system as instructed. Do not move or rename files unless explicitly told to do so in the instructions. Otherwise, you risk that some files will not be found, and you will end up getting no marks for this question or part.
5. During the exam you may not communicate with anyone, in any way. Any attempt to do so will be dealt with under the rules for Misconduct in Examinations.
6. Leave all mobile phones, MPEG players, pagers, PDAs and any secret communication devices in your bag at the door of the room.
7. You may not use the printers, CD-ROM drive or USB flash drives.
8. You must sit at the machine allocated to you.

Advice

1. You will have no internet access, however the following web resources have been copied onto the local file system:

- The class web site is at
`/dept/dcs/comp2100/public/www/index.html`
- The full Java JDK and API documentations are at
`/dept/dcs/comp2100/public/www/jdk/index.html`
`/dept/dcs/comp2100/public/www/jdk/api/index.html`
- The full Java Tutorial and Swing Tutorial are at
`/dept/dcs/comp2100/public/www/tutorial/index.html`
`/dept/dcs/comp2100/public/www/tutorial/uiswing/index.html`
- The Apache Ant Tutorial is at
`/dept/dcs/comp2100/public/www/ant/index.html`
- The Advanced Bash Scripting Guide is at
`/dept/dcs/comp2100/public/www/abs-guide/HTML/index.html`

As the convenience measure, these links are available on the toolbar of your web browser. Unfortunately, the links *between* these resource (Java tutorial, Java Swing tutorial and Java API) won't work, but the internal links will.

2. Each part has an estimate of its difficulty at the top. It is probably a good idea to do the easier part first. The marks for the different parts do not necessarily reflect the amount of work, so you need to pay attention to the time and not waste too much time on something small (markwise) but difficult.
3. If you mess up something and want to start anew, there is a local copy of all the files you were given available at `/dept/dcs/comp2100/exam`
4. Part marks are possible for all questions. Just because the requirements say "must" does not mean you've failed if your program doesn't do that. Do the best you can, and don't panic.
5. Good Luck!

QUESTION 1 [25 marks]

Bash scripts.

- (a) A lecturer wants to send email to all students who have not submitted an assignment. Your task is to write a Bash script that will prepare a list of the names and email addresses of all students who have not yet submitted the assignment.

You have two data files to work with.

The first is a plain text database file that lists the name, lab group and student ID of every student in the class. The file has one line for each student. The format of the lines is:

```
student-id      lab-group      Last-name, Other names
```

There is a single tab character between the student-ID and the lab group, and another tab character between the lab group and the family name. There is a comma followed by a space between the family name and the other names. An example file `database.txt` satisfying these conditions is in the `q1` directory.

The second file is a log file that has a line for every time a student submits the assignment. The format of the lines is:

```
student-id      time & date of submission      filename
```

There is a single TAB character between the student-ID and the submission time & date, and another TAB character between the time & date and the filename.

Write a Bash script that will print a comma-separated list of the names and email addresses of students who have *not* submitted any files, in a format suitable for cutting and pasting into an email program.

The script must be called 'rogues'. It must take the names of the database file and the log file as its two command line arguments.

Here is an example interaction with the script:

```
[comp2100@partch]$ cat database.txt
u4567890      wed09-11      BARNES, Ian
u2345678      tue13-15      WALKER, Richard
u1234567      wed11-13      JARSO, Tamiru
u3456789      tue17-19      KHOREV, Alexei
[comp2100@partch]$ cat log.txt
u1234567      13 June 2005 19:54:38  a2.jar
u4567890      13 June 2005 22:37:04  a2.jar
u1234567      14 June 2005 08:11:53  a2.jar
[comp2100@partch]$ ./rogues database.txt log.txt
Richard WALKER <u2345678@anu.edu.au>, Alexei KHOREV <u3456789@anu.edu.au>
```

Note that there must be no comma after the last name in the list.

Hint: You may find it useful to look up the manual pages for the `cut` and `grep` commands.

Important: Your solution *must* be in an executable file called `rogues` in the `q1` directory. I recommend that you test your script on the sample files *and* on other test data.

[15 marks]

(b) A little harder...

For loading marks into Streams, the lecturer needs a text file with comma-separated values (a CSV file) with one line for each student who has submitted the assignment.

Write another Bash script called `csv` that prepares a template file for the lecturer to enter marks into. Using the same two data files as command-line arguments, this script must write one line to the standard output for each student who *has* submitted the assignment.

The format of each output line must be as follows: first the student-ID, then two commas (the lecturer will enter the mark between them), then the student's name, another comma, and finally the date and time of the student's last submission, in the format year-month-day hour:minutes:seconds timezone-offset as in the example below.

The lines of output must be sorted into ascending order of student-ID.

With the same data in `database.txt` and `log.txt` as in part (a), here is an example interaction with the script:

```
[comp2100@partch]$ ./csv database.txt log.txt
u1234567,,Tamiru JARSO,2005-06-14 08:11:53 +1000
u4567890,,Ian BARNES,2005-06-13 22:37:04 +1000
```

Note that the spacing must be *exactly* as in the sample output above. In particular, there must be no space character between the comma and the start of the name.

You may rely on the lines in the log file being sorted in chronological order (earlier submissions will appear before later submissions in the file). You cannot rely on entries in the database file being in order of student-ID.

Hint: You may find it useful to look up the manual pages for the `sort` and `date` commands.

Important: Your solution *must* be in an executable file called `csv` in the `q1` directory. I recommend that you test your script on the sample files *and* on other test data.

[10 marks]

QUESTION 2 [25 marks]

This question is a modification of Homework 7 (Numbers to words II: Big numbers).

In the q2 directory you will find a java program `Words.java`. This program is partially completed. It contains:

- a `main()` method that reads the user's input from the command line;
- declarations for three `hashMaps` that give you easy access to all the strings you will need, together with a static block to initialise them;
- a working implementation of the `threeDigitsAsWords()` method that converts a number in the range 1–999 to words; and
- a *stub* for the missing `integerAsWords()` method that does the actual conversion.

Your task is to fill in the body of `integerAsWords()` so that the program does the required conversion. You will almost certainly want to call `threeDigitsAsWords()` at various points.

Modify this program so that it can handle *any legal Java integer*. Note that in Homework 7 this was restricted to *positive* integers. For this exam your finished program must be able to handle zero and negative integers also.

Here are modified extracts from the original homework requirements. Some of these requirements are already satisfied by the code I have given you.

The program shall take exactly one command line argument, which should be an integer. If it gets the wrong number of arguments or if its argument isn't an integer it shall print a usage message and exit.

If the input is OK, it shall print out the value of its argument, written out in words.

For example:

```
$ java Words 17
seventeen
$ java Words 123
one hundred and twenty-three
$ java Words 40
forty
$ java Words 2001
two thousand and one
$ java Words 20000001
twenty million and one
$ java Words 200030017
two hundred million, thirty thousand and seventeen
```

```
$ java Words 2147483647
two billion, one hundred and forty-seven million, four hundred
and eighty-three thousand, six hundred and forty-seven
$ java Words 0
zero
$ java Words -1
minus one
$ java Words -2147483647
minus two billion, one hundred and forty-seven million, four
hundred and eighty-three thousand, six hundred and forty-seven
```

Output should be formatted exactly as in the example above.

Notice the use of the word “and” in the output. This program must write numbers in the Australian/British style (“one hundred *and* thirty-seven”), not the American style (“one hundred thirty-seven”).

Make sure you test your program carefully. You will be penalised if your program crashes when we test it. You will be penalised severely if it fails to compile.

Hints: Start by solving the original homework problem, that is, by only considering positive integers. Then when you have that working, think about how to extend it to zero and negative integers as well.

Use the `threeDigitsAsWords()` method in the implementation of `integerAsWords()`. Roughly what you have to do is break the number up into three-digit groups (the billions, the millions, the thousands and the rest), and use `threeDigitsAsWords()` to print each of these, followed by “billion”, “million” or “thousand”, with appropriate use of commas or “and” in between. It’s the details of exactly when to use a comma and when not to, when to put “and” and when not to, when to leave something out etc. that makes this a bit tricky.

Important: Your solution *must* be in a file called `Words.java` in the `q2` directory.

[25 marks]

QUESTION 3 [25 marks]

Ethics, Ant build script and the project

(a) Professional ethics.

Consider the following scenario:

Jane is a member of the IEEE Computer Society and the Association for Computing Machinery. She works as a software developer for an internet company that creates peer-to-peer file sharing software. Although it has legal uses, this software is mostly used to distribute illegal copies of music, feature films and other proprietary content. The Hollywood studios and major record labels have been trying to stop this by tracking network file transfers and using this evidence to take individual users of the software to court for copyright violation.

To avoid these attacks, developers are working on an innovative cyclic encryption of the data streams that will prevent people listening on the network from proving any illegal activity. Jane is asked to help with this work.

Unconfirmed reports suggest that the software is being used for secure, untraceable distribution of child pornography. Another report suggests that terrorist groups are using the software. Yet another report says that organised crime groups are using it to co-ordinate drug deals.

Two officers of the Federal Police approach Jane and ask for her assistance in providing a back-door to the software that will enable them to catch and convict users engaged in criminal activity. They say that they are not interested in copyright violations, only serious crimes. They threaten Jane with criminal charges if she refuses to co-operate or tells the other developers about this.

Discuss the issues raised by this scenario, in the light of the software professional's two main areas of professional responsibility:

- To his or her employer; and
- To the broader public interest.

Note that you are *not* asked to say what Jane should do, but rather to discuss the various factors she should take into account in making her decision.

Important: Write your answer in plain text in a file called `Ethics.txt` in the `q3` directory.

[9 marks]

(b) Ant build script.

This question is based on Exercise 5 of Lab 3 and exercises 1–6 of Lab 6, and assumes basic knowledge of writing Ant build scripts (Lecture 12).

Go to the q3 directory. In there you will find the following source files:

- IntegerArray.java
- ArrayTester.java
- integer_array.c
- timer.c
- Timer.java
- Makefile

Write an Ant build.xml build script that automates the process of compiling and testing this program. You may use the Makefile as a template, meaning that what Makefile does, the build script will also be able to do.

When the user types ‘ant -p’ at the command line, the output should be the list of targets defined in the build script build.xml. These targets should be run, compile, make-h, compile-c and clean. The project name should be lab6, the default target should be run. When the user types ant, it should build all the required files and run the test.

Hints: The C compilation task which is captured by the compile-c target, should be defined using the exec task to invoke native executable gcc. You may find it useful to use the Apache Ant Manual which is among the provided on-line materials.

Important: Your answer must be in a file called build.xml in the q3 directory. Be careful that the clean task does not delete your solutions to parts (a) and (c).

[9 marks]

(c) Reflection on the project

The project software which was used in our assignments is only a prototype designed for educational purposes, and some aspects of its design that could be improved. Suppose you were given the task of redesigning and rebuilding the project software (the endpoint of Assignment 2: the graphical interface document browser) with a view to releasing it as commercial software.

What changes would you make to the design? Why?

Hint: This is *not* about changing or adding to the requirements, but about changing the internal design and construction of the program.

Important: Write your answer in plain text in a file called Project.txt in the q3 directory.

[7 marks]

QUESTION 4 [25 marks]

Graphical user interfaces with Java Swing.

Go to the q4 directory, where you will find the Java source code for a modified version of the Clock application from Lab 4. It retains the Digital Clock only (the Analog Clock is gone), with a few additional menu features which allow the user to choose between the time display or the date display. It also has an option to show or hide a label with the alarm time and mode. (The original Model has been extended to include the alarm time and mode.) In this question you will add some new features to it.

(a) Straightforward.

Extend the code to allow the user to choose between 12 and 24 hour time representation. The 12 hour display must include “AM” or “PM”. The Digital Clock display must be placed inside a `JTabbedPane` with the tab buttons for choosing the desired representation. When completed the application window should look something like in Figure 1:

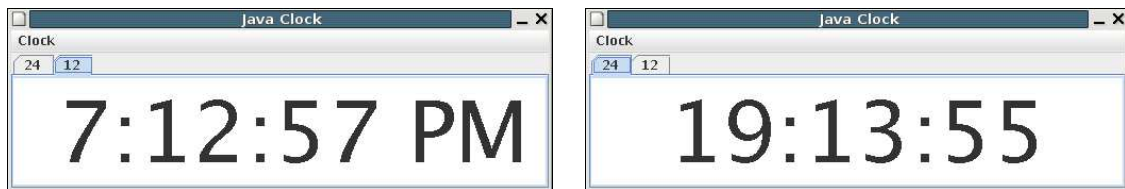


Figure 1: The 12 and 24 hour display modes. Note the tabs for selecting the different modes.

Hint: You will probably want to look up how to create a `JTabbedPane` and add `JComponents` to it in the API documentation and/or the Swing tutorial. The model already contains all the required elements to set the time representation to either 12 or 24 hour formats.

[10 marks]

(b) A little harder.

Add a new item in the Clock menu called ‘Set Alarm’. When selected, this should pop up a dialog box that asks for a selection ‘ON’ or ‘OFF’. When the user clicks ‘OK’, the dialog box should be dismissed, and the value displayed on the ‘Show Alarm’ label should be modified accordingly.

Note: You do *not* need to change the Alarm time, only the Alarm mode (ON/OFF). Make sure the dialog box displays with the standard question-mark icon, as in the screen shot in Figure 2.

Hints: You will probably need to look this up in the Swing tutorial.

[8 marks]

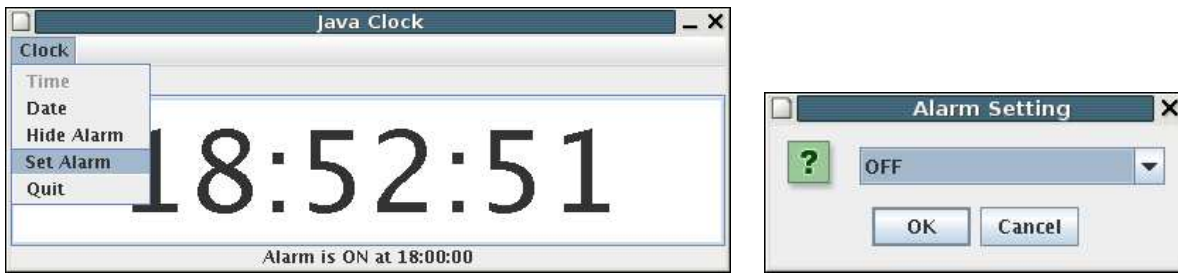


Figure 2: The new “Set Alarm” menu button and the dialog box.

(c) Harder still.

In the original code, once either the Time or Date representation is chosen using the Clock menu, it remains unchanged indefinitely unless the user makes another menu selection. Change the original code such that the Date representation, when chosen, lasts only 5 seconds (assuming that the user does nothing during this interval), after which it is automatically switched back to the Time representation, as demonstrated in Fig. 3.

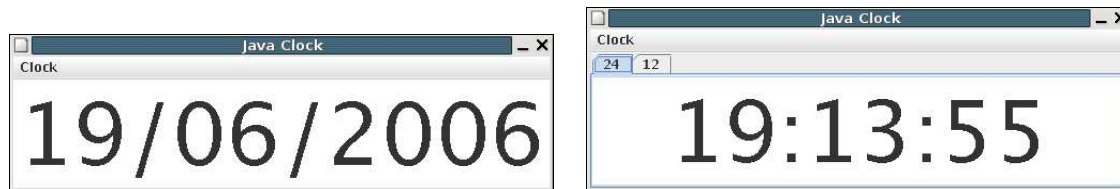


Figure 3: Left: Immediately after selecting the Date menu item. Right: Five seconds later.

Hint: You will need to look up the `Timer` class in the API docs and/or the Swing tutorial.

[7 marks]

Important: Your solutions to the Question 4 must be in the Java source files in the `q4` directory. Since all three parts involve modifying the same code, take care that what you do for the later parts doesn't mess up what you have done for the earlier parts. I recommend that you test all three parts once you have finished.

Checklist:

Before you leave the exam, make sure that all your answers are in the right files in the right locations.

- Your answer to Question 1 part (a) must be in the Bash script `q1/rogues`.
- Your answer to Question 1 part (b) must be in the Bash scripts `q1/csv`.
- Your answer to Question 2 must be in the file `q2/Words.java`.
- Your answer to Question 3 part (a) must be in a plain text file `q3/Ethics.txt`.
- Your answer to Question 3 part (b) must be in `q3/build.xml`.
- Your answer to Question 3 part (c) must be in `q3/Project.txt`.
- Your answer to Question 4 parts (a), (b) and (c) must be in the various Java source files in the `q4` directory.

