Introduction to Programming

Week 3 tutor’s kit

Your preparation for this week

- You should have skimmed the first section of Kingston, taking more notice of the ~80 pages that are the Blue material - and be aware of what is in the remaining 4 chapters.
- Read these notes and the associated support material in Section 5.6 (pp 47 -- 51) of the COMP1001/1901 Resource Book.
- You should have a way of filing things for each group and for each student in your class. Each week, I recommend that each group use the on-line software to produce a plan. You should keep this filed for that group and review it with the group. If they use the software, they can easily print a copy of the plan for you (and have copies for themselves). Each student should be writing some code each week. They should hand you a copy of that code PLUS a plan for what they will do in the coming week. Clearly the student will need a personal copy. The reason they should also write/print a copy for you is that it constitutes a greater form of commitment.
- Make sure you have the Tax Bracket problem from the web site (and also available on staff at judy/IP/3_task_page.cgi)
- Please try taking a half hour break from the lab this week. You should then claim 2.5 hrs lab time and 1.5 hrs preparation time, since the preparation load is little high at this stage of the semester. Next week, tutors will cut back to being in the lab for just 1 of the three hours for teaching (+ marking time for Problem 1, 5 minutes per student for the demo + 5 minutes per script to mark).

1. The situation

As this is week 3 and Problem 1 is due in week 4, students must be heading towards writing some reasonable code this week. There is a good chance that students will be feeling a sense of panic. Your job is to calm them and to help them plan how they will proceed.

Note that people who do not make the week 4 deadline will need to continue working on their problem 1 till it is ok. The important thing is that they be making progress towards the goal. See some tips in the workshop below.

Some students are unclear of the role of the groups. They should be encouraged to focus on the assessment criteria for Problem 1 as these make it quite clear that their individual code is what is assessed.

As stated in section 2.3 of the COMP1001/1901 Resource Book, seminar themes are statements and lists, if, loop, list traversal. Prac themes are self-assessment in various ways, with the above concepts as a focus point.

2. Tutorial script

<table>
<thead>
<tr>
<th>Activity 1</th>
<th>time for it minutes</th>
<th>total time minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline tutorial:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do this from the details below.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activity 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concept inventory:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COM1001/1901 Tutor Kit Semester 1 2000
This task uses section 5.5, pages 44-46 of the COMP1001/1901 Resource Book.
Note that the concept inventory should help students see how far they have come in two weeks since they will have some familiarity with most of these concepts.
It should be done individually. Students should have the time to quickly read the material on page 44 and ask questions about it.
Then should fill in the bottom of page 44 and quite quickly the ‘Score’ column on pp 45-6. You should walk around while they do this so that you can see how people assess themselves on these things.

Ask students to fill in the ‘Notes on what to do about this aspect’ in time for the next lab class so that it can feed into their planning their work for the lab. The sorts of things that should go here depend on the type of aspect and their current level of knowledge. They need to identify current priority aspects to learn, ie things that they think are important so these will appear on plans. They need to establish which require reading books, which require working at the Blue workbench - and if they have read the assessment criteria for Problem 1 (page 21 of COMP1001/1901 Resource Book), they should mark the aspects needed for that as higher priority.

Bring the class together to get an idea of what is generally known. What is generally not.

Note: Many students will not know a concept like ‘control structure’
They will know ‘if’ and ‘loop’ (a bit)
You can tell them about meanings of such terms
Or, if time permits, ask them to see if they can use the books to work out what ‘control structure’ means.

<table>
<thead>
<tr>
<th>Activity 3</th>
<th>time for it</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-assessment:</td>
<td>minutes</td>
<td>minutes</td>
</tr>
<tr>
<td>This task uses the ‘Tax bracket’ task described in the attached screen dump from the self-assessment web site.</td>
<td>30</td>
<td>55</td>
</tr>
<tr>
<td>Ask students to work individually on this task. Walk around seeing how they are coping. Then ask them to form groups of 2-3 to continue working. Ask them to discuss different ways to do it. (You might hint about case, v if.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explain that they need to learn to assess their own code. To do this, they need to know what criteria to apply. Ask them to evaluate their group solution using the following criteria (essentially those at the self-assessment site):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Routine Header: with a good name indicating what it does eg. TaxBracket have an integer input parameter with sensible name eg. income comment stating what routine does</td>
<td>15</td>
<td>70</td>
</tr>
<tr>
<td>Precondition: ensures that input parameter meets assumptions of your code (and if no assumptions needed, true is fine)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Postcondition:
captures the effect of your code
(since this is writing, true is fine with a comment after it)

Style:
indentation - always consistent
use constants for the boundary values, 5000, 8320, 33222

Control flow and testing
exactly four possible routes through the code
correct on 450 - check out cases you have been told answers for!!
correct on 15,000 and other normal values well within the ranges
the exact boundaries should be correct, ie. 0, 5000, 8320, 33222
testing was systematic - ie each test was purposeful
(you could explain to someone why you did each)

Ask them to look at the evaluations and discuss why we include them. 10 80
Ask them to do this in groups first and then share with the class.

<table>
<thead>
<tr>
<th>Activity 4</th>
<th>time for it minutes</th>
<th>total time minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review what people have read.</td>
<td>15</td>
<td>95</td>
</tr>
</tbody>
</table>

Ask people to say what they have read and summarise it on the board.
Ask them to say how helpful it was - and allow people to suggest things they would like clarified because they had trouble understanding them. Make sure you tell them that they should be using this activity to help them add to their plan for the coming week.

Things that should come up (with you prompting if need be) are:

• Kingston book chapter 1 - when this comes up, please ask if they did the exercises at the end - help them explain why doing the exercises is a good idea (viz that is they do not do the exercises they cannot know if they really did understand). I would estimate this chapter would take about 1 hour to read carefully and make a go at the exercises. Ask them to estimate how long it took. Let your section leader know.

• Kingston book chapter 2.1-2.4 - this is stuff is incredibly useful for Problem 1. Again ask if they did the exercises - they would have needed to come and use the machine on their own time to do some of them but I bet they would have learnt heaps by doing the exercises. Ask if the people who have read this found it useful for Problem 1. Try to get students who have done this reading to tell the class about it so that they can inspire others to do it.

Note that the rest of Chapter 2 is too hard at this stage - it is better if people can come up with the conclusion that this was so. I would estimate that it would take about 1 hour to read this material and about 2 to do the exercises.

• Kingston book chapter 3 - I think this should be too heavy at this stage but the headings should have looked relevant since there are various types of variables in the programs. So this is a chapter that they should have decided to skim, picking out bits that are understandable now and making a note to themselves to return to it before the end of the semester.

They will really like Kingston chapter 4 - commend it to them.
Try to get people who tried to read it to volunteer this sort of thing.
Students really listen to each other on this sort of thing. (PS. Let your Section leader know if this is your experience.)

- COMP1001/1901 resource Book by Kay - they should really have read sections 1, 2 and 3 in great detail plus have mastered the Unix and Blue in 6.1 (Getting started with Unix)

<table>
<thead>
<tr>
<th>Activity 5</th>
<th>time for it</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>You decide</td>
<td>minutes</td>
<td>minutes</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>110</td>
</tr>
</tbody>
</table>

- You decide

Ideally, there will be things that came up in the class. Here are some other possibilities:

- The code in http://www.ug.cs.usyd.edu.au/~cs1/blu/random/Demo.blu which is reproduced at the end of these notes. You could put it up on the board for them to read. See if they can work out what it does. This example gives you the opportunity to mention library classes. It also gives you a chance to help them see that they can guess what things do. And that there are good things to be found among the examples.

Ask them to use it to print random numbers in the range 1..100. Or to think how the bank could use this for a lottery - so that lucky customers could get a free lottery ticket with each deposit and you could have a draw.

Part of the purpose of this example is to let them see that the Problem 1 tasks are not constrained. They are welcome to do weird and creative things: perhaps a teller who plays the game scissors, rock, paper; or this lottery; or anything else they think up as relevant to the task as stated.

- The code from http://www.ug.cs.usyd.edu.au/~cs1/blu/timer/timer.blu a timer class, something that might interest them. You might invite discussion of the possibilities of day ‘timer’ that simply increments a day at a time (since bank interest, as far as I know, never operates at finer resolution than daily).

- Suggest they write a Teller class co-operatively where the Teller asks the customer if they are over 70 and if so, asks them how much they want to deposit and if it is over $1,000, they get an automatic bonus of $50.

Note: you set each task for the class, let them work in groups on it while you walk around seeing how they go. Invite one of the groups to present their answers to the class.

The idea is to see if you can step off that stage at the front of the class and let the students do much of the talking, with you encouraging and ensuring that things are correct. You might also summarise at the end of each problem, highlighting things that are important.

It is better to do fewer of these examples, but thoroughly.

- you could similarly have students work through the chapter 1 exercises if people had trouble with them eg some of my people found swapping hard - a group who wants to work on that could sit together
3. 3-hr Workshop

Although the students are supposed to plan their own workshop activities, the following gives some recommended activities for this week. This is supposed to help consolidate their understanding of how to use this time to work together in learning new things.

It is important to emphasise (in your introduction) that a major reason for working in groups as they learn new aspects of Blue is that a person working alone can work out less than that same person can do when working with others, talking to them about what they are doing, about difficulties and sharing the jobs of looking up resources.

This workshop, the practical activities are intended to illustrate how they can set the goal of exploring next concepts in the lab and getting confidence by writing code, co-operatively.

You will need to ensure that people are not working alone: each group should split into subgroups, each of 2-3 students on the task that they have chosen.

As a break from the pure Blue, the whole groups should come together to discuss the group plan.

<table>
<thead>
<tr>
<th>Activity 1</th>
<th>time for it</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class overview</td>
<td>minutes</td>
<td>minutes</td>
</tr>
<tr>
<td>Construct from the details below</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Activity 2</th>
<th>time for it</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>LList and self-assessment site</td>
<td>minutes</td>
<td>minutes</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>75</td>
</tr>
</tbody>
</table>

They should treat this as an experiment in one approach to using the self-assessment site resources. Ask them to suppose that they had set the goal of learning about LLists. They should consult a range of resources, but here we focus on:

- The Kingston book
- The Kollar book

Meanwhile you go around and collect the evidence of work done in the preceding week, hopefully, spending a few moments on how it matches what they had planned to do.

Tick it off on your marksheet.

**This means checking each student.**

You should also collect group plans for the coming week.

It is fine for groups to hear you comment on each other’s plans.

Be sure to give clear feedback. For example, I found myself writing things like ‘good overall goal’, ‘plans still a bit vague’, ‘more concrete deliverables’, ‘need to plan timing’, ‘need stronger statement of how to judge success’.

<table>
<thead>
<tr>
<th>Activity 3</th>
<th>time for it</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pause gymnastics</td>
<td>minutes</td>
<td>minutes</td>
</tr>
<tr>
<td>perhaps neck stretches?</td>
<td>10</td>
<td>85</td>
</tr>
</tbody>
</table>

COMP1001/1901 Tutor Kit

Semester 1 2000
Activity 4

Your overview of this activity will need to emphasise that you want them to use this part of the class to practice:
• incremental development
• construction of a complete class from scratch
• co-operative learning in class

Ask students to write a very simple class written incrementally, from scratch - if they have a simple class in mind that is great but you should provide an example of one: a class to represent a student, who has a name, sid and login, and where you can set these and get them.
• Encourage students to consult the Account class
• Encourage them to write just the header and make sure that compiles And runs (doing nothing).
• Then they should put variable declarations in, compile, run.
• Then the two routines to set up just the name, compile, run.

and so on.

Finally, your Section Leader will appreciate mail on each of these activities: what was known in the concept inventories, how much reading they have done, what code they could write in the tute, how the self-assessment site worked, the pause gymnastics and the incremental development. Also pass on a status report on how the groups are operating and the feel/tone of the class.

Judy Kay
2000 PBL Co-ordinator for Comp1001/1901
March 2000
Code examples for tutorial activity 5

Time example from cs1 web site

You might make this an overhead slide if your tute room has a projector. If not, you can write the exciting parts on the board as they do the earlier part of the class.

class timer is

==============================================================================
== Author: mik
== Version: 1.0
== Date: 31. July 1996
== Short: Simple timer for hours and minutes
==
== Timer implements a timer object that can hold and manipulate time values
== consisting of hours and minutes. The time can be set, incremented and
== displayed on the standard terminal.
==
==============================================================================

uses display

internal

var
  hours: display
  minutes: display

interface

creation is
  == Create a timer with hours and minutes display. The initial
  == time is 00:00.
  do
    hours := create display (24)
    minutes := create display (60)
  end creation

routines

increment is
  == Increment the timer by one minute.
  do
    minutes.increment
    if minutes.get_value = 0 then
      hours.increment
    end if
  end increment
set (hour: Integer, minute: Integer) -> (done: Boolean) is
  == Set the timer. A check is made whether the provided values are
  == legal. If they are, the time is set and "true" is returned, if
  == not "false" is returned and the time remains unchanged.
  do
    done := false
    if hour<0 or hour >= hours.get_limit then
      return
    end if
    if minute<0 or minute >= minutes.get_limit then
      return
    end if
    hours.set_value (hour)
    minutes.set_value (minute)
    done := true
  end set

show is
  == Display the timer on the standard terminal.
  do
    hours.show
    print (":")
    minutes.show
    print ("\n")
  end show

end class
Example of random number code

class Demo is

==============================================================================
== Author: Michael Koelling
== Version: 1.0
== Date: 2 January 1997
== Short: Demo class to show use of random numbers
==
== This class shows the use of random numbers. It provides two routines
== that use a "Random" object to generate numbers.
==
==============================================================================

uses Random

internal

var
  rand: Random -- the randomiser

interface

creation is
  == Create the random number demo.
  -- Here, we create a randomiser object (of class "Random"). This
  -- object will be used subsequently to create random numbers.
  do
    rand := create Random (nil)
  end creation

routines

print_numbers is
  == Print some random numbers.
  var
    cnt: Integer
  do
    print ("How many number do you want? ")
    cnt := input.readInt
    loop
        exit on cnt = 0
        print (rand.nextInt (100), "\n")
        cnt := cnt-1
    end loop
  end print_numbers