Introduction to Programming

Week 2 tutor’s kit

1. The situation

- You should expect changes in the class this week still, some leaving and others coming. You should try to follow up people who attended last week but do not come this week (eg you could check if they have logged in using finger login-id)

- Students are only in your class if their printed timetable says so. In due course this will match the information that the command markssheet gives.

- Students without programming experience tend to feel insecure at this stage. Please try to identify such students and make sure you know how they are doing. Let your section leader know this.

- Students will be feeling uncomfortable with the planning we ask them to do. You will need to encourage them. It is helpful if those who have done the planning are treated as though they are doing as expected (and others should be told that they really must hand in plans to you).

- Students should have written their own piece of Blue code to do something by the end of this week. If you find someone flying ahead, make sure they are aware of the possibility of moving to the 1901 (Adv) from 1001 and to the SSP option in 1901.

- This week we start making a file for each student. The best way to do this is to get a Lever-Arch file and separators or plastic sleeves and keep printouts each for each student filed in their section.

2. Tutorial script

Note that students will be keener to do the technical Blue stuff than the generic problem solving activities. To ensure that both get done, they are integrated so that the generic problem solving issues are explored in terms of Blue and learning it.

<table>
<thead>
<tr>
<th>Activity 1</th>
<th>time for it minutes</th>
<th>total time minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awareness of problem solving style, listening:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This task is described in section 5.3, page 40 of the Comp1001/1901 Resource Book.

1.1 Introduce task and ask students to form pairs (or threes as the Book says).

The thing to emphasise to your class is that this is a problem solving activity, not an exercise. So, we expect they will find the six tasks (starting on page 40) are difficult, non-obvious. They will need to get out resource books and guess and even decide there are some things they find too hard.

Unless you emphasise that this is meant to be hard, students may be very discouraged.

(Postcondition: they know the goal of this task and how to do it)
1.2 Allow students to take turns being the problem solver and listener and to work through the problems

1.3 Reflection on problem solving style and listening skill. They should fill in the questionnaire on pages 41/2 and discuss them with their partner.

1.4 Individually complete the two free form questions on the lower half of page 42 - this is the chance for them to think quietly about themselves and what they learnt about themselves from this activity.

1.5 Group discussion on things that came up that they found really difficult in this task, from the point of view of Blue.

Note that this activity had two aspects
• learning about generic problems and oneself - and you need to emphasise this - if they get the hint from the activity, that is excellent - if they do not, there will be other opportunities
• the point of view of Blue - the things that they found hard - and they will typically have had trouble using the resources to work out some aspects. Your task here is to consolidate the most important aspects of the task: the overall flow of control, the overall action of the code, the way they can use printed resources and each other to work out technical things as they have difficulty with them.

Post: they will appreciate which aspects of these examples are important to understand now and which they should be looking up in books.

They will realise this was a listening and introspection activity.

They will know these were supposed to be hard examples.

---

**Activity 2**

**Problem solving skills: tracing and generalisation**

2.1 Introduce activity
• This activity is taken from section 5.4 of the Comp1001/1901 Resource Book. (Page 43)
• As usual introduce this activity. It is done in their groups. You might let them read the task and see if they can work out what they have to do.

2.2 Let them start but make sure you walk around giving help. I imagine that most groups will be able to get started but may not fill in the tracing table suggested at the bottom. I think it is better to help with this by seeing each group, rather than show the whole class at the board - the latter approach encourages them to wait for you to tell them things which we do not want.

By walking around to each group, you can gently push them towards tracing the code systematically by asking questions like: what happens at the first step? can you draw up a table like the one at the bottom of the page for each step.

2.3 Generalisation step as class activity.
• As a class they should be asked to help write the missing comment on the code.
• Ask them about generalisation and why it might be important for loops.
• Ask them to suggest how tracing can help with generalisation. (ie. it gives you some examples to help you see the pattern)
• Ask them to suggest when tracing is a good thing to do (ie when you find the code tricky and have to really think about it by doing the same steps the computer does)
• Note this is an important strategy for debugging.
• Might suggest they look at the Blue facilities for single stepping since they can then check out exactly what happens in code they find mysterious. (The Kolling book describes this - perhaps one person in each group could undertake to really work through all the stuff about the Blue environment in Kolling over the next week.
• Ask them the limitations of tracing (ie you need to generalise from it to be able to say what the code really does in general - and that can be difficult for beginners)
Note that this example was purposely written so that it is hard to work out what it does - everyone will need to read it carefully and tracing should help most people work it out
You might point out that any time their code is as hard to read as this, they should think carefully about ways to do things more simply

3. 3-hr Workshop

<table>
<thead>
<tr>
<th>Activity 1</th>
<th>time for it minutes</th>
<th>total time minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class overview</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

Every class starts with an overview of the main activities.

This time it should be working on aspects of Problem 1. You will have to make sure that they have ideas for group work as it is very easy for them to want to work individually and that is not what happens in the workshop.

You should ask the class to help you construct a list of things that they plan to do in this class. This means that each group can help the whole class get a good set of ideas.

This plan should include:

• trying to write at least one small Blue program from scratch. A good starting task is a Human-Teller class which asks the user how much they want to deposit/withdraw.

• making sure that each group member is on top of the material in the Getting Started tutorial that was begun last week -- if they did not finish it in the last workshop, they should have done it in their 6 hours of private work each week.

• explore the code of some of the examples in `cs1/examples - let them snoop around to find the Problem 1 directory (using the unix notes)

<table>
<thead>
<tr>
<th>Activity 2</th>
<th>time for it minutes</th>
<th>total time minutes</th>
</tr>
</thead>
</table>
They do their stuff

Meanwhile you go around and see each student’s plan for the coming week and tick it off on your marksheet. If they are to do the planning properly, you will have to be very serious about the need for it. At the same time, collect the evidence of work that they did in the preceding week. Put both the plan for the coming week and the evidence of work done in the last week into your file for that student.

If they are to cope well with problem 1, they MUST plan.

<table>
<thead>
<tr>
<th>Activity 3</th>
<th>time for it</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pause gymnastics - as last week - or even more :-)</td>
<td>10 minutes</td>
<td>85 minutes</td>
</tr>
</tbody>
</table>

Make sure people take some break in the 3-hrs
We want them thinking hard while they are there

<table>
<thead>
<tr>
<th>Activity 4</th>
<th>time for it</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>minutes</td>
<td>minutes</td>
</tr>
</tbody>
</table>

They continue working and you walk around helping upon request, ensuring you see every student in each class

For classes ending at 6pm
Close all windows
Lock the door to your lab
Finally, remember to mail feedback, comments and requests for seminar content to your section leader.

Judy Kay
2000 PBL Co-ordinator for Comp1001/1901
March 2000

Trouble shooting:
- `reduce` is a command you can use to get students back to the default unix environment, useful if their friends install some funny environment that swirls or whatever.
- `grep /etc/passwd` for student details and use `passwd` command as needed to fix passwords.
- `quota` tell you if the student is over quota
- `search-and-destroy` searches their file space for redundant files (such as "core") and removes them with (y/n) for doubtful large ones.
- If Blue colours are wrong, it is probably because Netscape has taken the colours. To fix, quit Netscape and start Blue again. (Blue actually works properly anyway - just looks wrong).
- For students who enrol late and do not appear in /etc/passwd, tell them to register at the Computer Science Help Desk, Madsen LG44, 12-2pm and 5-5.45pm daily. Their account should be created within a day or two of registering.