Informatics 1: Object Oriented Programming

Tutorial 01

Week 3: Pair Programming 27/01 - 31/01

Volker Seeker (volker.seeker@ed.ac.uk)
Vidminas Mikucionis (vmikucio@ed.ac.uk)

1 Introduction

In this tutorial, we will practice working on algorithm problems. These are the kinds of questions that showcase fundamental concepts of programming (and will be covered in much more detail in second year, if you take the algorithms & data structures course). These concepts are often applicable to many kinds of problems and will likely help you with future courses, projects and programming career.

The tutorial workshops for the Inf1B course are designed to support collaborative learning and programming in groups of two or more people. Programming "in the wild" is rarely done by individuals alone. Even if you are working as a self-employed freelancer, you will likely share sources and libraries with other developers. It is therefore an essential skill to learn how to communicate with others about code, how to work with and extend code written by others and how to share programming tasks amongst multiple people.

We will start this first workshop with a technique that is commonly used in industry called *pair programming*.

☞ While you are welcome to read through the tutorial sheet beforehand, you are not required to prepare the exercises at home before the tutorial. The tutorials are designed as workshops which you work through during your tutorial session. As preparation work, please focus on keeping up with labs and lectures instead.

2 Warmup

Task 1 - Team forming time! ◁ Task

You will already have seen many programs before, both in last semester and in the lectures leading up to this tutorial. However, there are many ways of looking at and understanding programs.

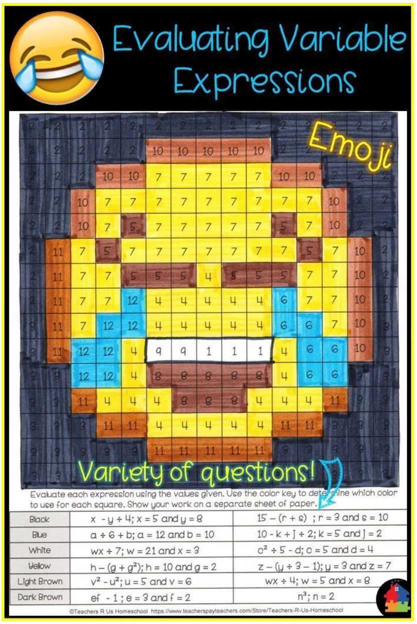
One such way is to imagine yourself in a role of a variable, keeping data as the program executes.

This can scale up to bigger programs with multiple variables too. To try this out in practice, form groups of 6 people first.

Task 2 - Manual walkthrough ◁ Task

Once you are in groups, take a look together at this piece of source code and choose the role of a variable each (if your group is fewer than 6 people, some people can take more than one):

```java
public class Obfuscation {
    public static void main(String[] args) {
        int t = 60, x = 22;

        if (t++ - 59 == 0)

        {

          x = 20;
        }

        int i = x; int j = 0;

        while (i --> 0) {
          i -= 3;
          j++;
        }

        int w = t + x - i * j;
        int k;

        for (k = 0; k < j; ++k) {t -= (k % 2);}
    }
}
```



Evaluate each expression using the values given. Use the color key to determine which color to use for each square. Show your work on a separate sheet of paper.

Black	$x - y \div 4$; $x = 5$ and $y = 8$	$15 - (r + s)$; $r = 3$ and $s = 10$
Blue	$a \div 6 + b$; $a = 12$ and $b = 10$	$10 - k + j \div 2$; $k = 5$ and $j = 2$
White	$wx \div 7$; $w = 21$ and $x = 3$	$c^2 \div 5 - d$; $c = 5$ and $d = 4$
Yellow	$h - (g + g^2)$; $h = 10$ and $g = 2$	$z - (y \div 3 - 1)$; $y = 3$ and $z = 7$
Light Brown	$v^2 - u^2$; $u = 5$ and $v = 6$	$wx \div 4$; $w = 5$ and $x = 8$
Dark Brown	$ef - 1$; $e = 3$ and $f = 2$	n^3; $n = 2$

©Teachers R Us Homeschool https://www.teacherspayteachers.com/Store/Teachers-R-Us-Homeschool

Figure 1: Fun Expressions[1]

What are the variables in this `main` function? What are their values at the end of the execution? You have about 10 minutes to figure this out.

Let each person in the group keep in mind only the value of "their" variable and talk together to figure out how each statements affects this.

This exercise is an example of a "manual walkthrough". In practice, going through code line-by-line with the values that variables hold (the program state) in mind can often help to identify sources of bugs.

3 Exercises

With this problem-solving technique up your sleeves, try having a go at designing a solution to some concrete programming problems (the kind that might even come up in an internship interview!)

You will have about 5 minutes to work on the exercises individually, after which you will have the opportunity to discuss ideas.

☞ In designing a solution, avoid writing any code - instead try to explain how it would work in the most intuitive way you can. Think using drawings, diagrams, comics, stories, bullet points, or anything else, be creative! We will do the actual implementation later.

Task 1 - Scrabble Winner

◁ **Task**

This problem is about the popular board game Scrabble. If you have never played it before - it is a game where players get pieces with letters that they use to form words in a crossword-like puzzle. Each letter is worth a different amount of points according to this table:

[1]Image source: https://i.pinimg.com/736x/c0/a8/39/c0a839ca9bb205c5471220b31132eef0.jpg

A is worth 1 — B is worth 3 — C is worth 3 — D is worth 2
E is worth 1 — F is worth 4 — G is worth 2 — H is worth 4
I is worth 1 — J is worth 8 — K is worth 5 — L is worth 1
M is worth 3 — N is worth 1 — O is worth 1 — P is worth 3
Q is worth 10 — R is worth 1 — S is worth 1 — T is worth 1
U is worth 1 — V is worth 4 — W is worth 4 — X is worth 8
Y is worth 4 — Z is worth 10

A word is worth the amount of points that its letters add up to.

For example, the word ″gear″ is worth 2 + 1 + 1 + 1 = 5 points.

Other possible word examples could be ″apple″ (9 points), ″queen″ (14 points), ″koala″ (9 points), ″object″ (17 points), etc.

Figure 2: Scrabble[2]

Try thinking of an algorithm that takes a word as input and outputs the number of points that the word is worth.

Hint: consider the input as an array of characters...

[Fun fact: In 2015 a man from New Zealand memorized every french word in the french scrabble dictionary and won the French Scrabble Championship. He still doesn't speak any french. https://www.npr.org/sections/thetwo-way/2015/07/21/424980378/winner-of-french-scrabble-title-does-not-speak-french]

Task 2 - Is it weekend yet?

◁ **Task**

[2]Image source: https://localscrabble.files.wordpress.com/2011/11/scrabble.jpg

Figure 3: Weekend Face[3]

Given a day of the week (as a string) and a integer number n, determine whether n days later is a weekend (Saturday or Sunday).

If so, print ”WEEKEND!” otherwise print ”not weekend yet...” to standard output.

For example, given the inputs ”Monday” and 2, your algorithm should figure that 2 days from Monday is Wednesday and so print ”not weekend yet...”

As another example, given ”Saturday” and 1, the next day after Saturday is Sunday, so your algorithm should output ”WEEKEND!”.

Task 3 - Optional: esrever

◁ **Task**

If you went through the first two exercises like a breeze, then this is an optional harder exercise to try.

Given a string containing a sentence, how would you output the sentence with the order of words in reverse? For example, given ”I like ice cream” the output should be ”cream ice like I”.

It may help to consider the given string as a character array.

Pair programming

Now that you have lots of ideas for the exercises above, try implementing the exercises in pairs. For this you will need to find a partner and share a computer between the two of you.

Once you have a partner, count the number of Scrabble points in each other’s name. The one whose name is worth more points will start in the role of the ”driver” and the other will be the ”navigator”.

As the driver, your task will be to write code while listening to advice and comments from the navigator. Your main job is to implement the program.

As the navigator, you will be looking at the program together with the driver. Keep the problem statement to hand, and help the driver stay on the right path by discussing what needs to be implemented and how. Your main job is to ensure the driver can implement the program.

Now you will have about 5 minutes to implement a solution to the first task. Then swap roles and try solving the second problem. If you don’t have a Java IDE installed, you can use this awesome online tool:
https://www.jdoodle.com/online-java-compiler/
If you were wondering how to go from a string to a character array - you can use the `toCharArray` function like this:

```java
        String sentence = "I like ice cream";
        char[] buffer = sentence.toCharArray();
```

Figure 4: Let’s get to driving![4]

[3]Image source: https://www.groundzeroweb.com/wp-content/uploads/2016/10/Funny-Weekend-Memes-8.jpg
[4]Image source: https://www.nickfreemansolicitors.co.uk/wp-content/uploads/2016/11/child-driving.jpg

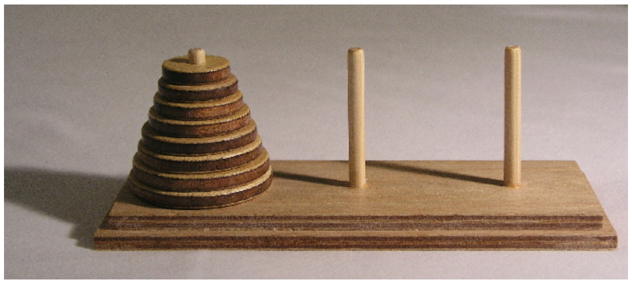


Figure 5: Tower of Hanoi[5]

4 Divide and conquer (the world)!

The Tower of Hanoi (also called the Tower of Brahma or Lucas' Tower) is a mathematical game or puzzle. It was invented by the French mathematician douard Lucas in 1883.

There is a story about an Indian temple which contains a large room with three old posts and 64 golden disks. Brahmin priests, acting out the command of an ancient prophecy, have been moving these disks for countless years. According to the legend, when the last move of the puzzle will be completed, the world will end!

Your task will be to find a different partner to pair up with and end the world together.

This puzzle consists of three rods and a number of disks of different sizes, which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape.

The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

- Only one disk can be moved at a time.
- Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.
- No larger disk may be placed on top of a smaller disk.

Simple algorithms exist for solutions involving three pegs, however, the optimal solution for the Tower of Hanoi problem with four or more pegs is still unknown!

You can familiarise yourself with the game by playing it here: `https://www.mathsisfun.com/games/towerofhanoi.html`

In pairs, try to solve this problem by first designing an algorithm on paper and them implementing it using pair programming as before. Your tutors will be there to help.

✎ **Hint**: You can start by designing an algorithm that solves this problem with only 3 pegs and 3 discs, and then extend it to cover more discs, until you get a solution that works for an arbitrary number of them.

Also, despite having an iterative solution, this problem is best approached using recursion.

Besides being an interesting mathematical game, this problem actually has some uses in the real world! Here is an article about some real world applications of the Towers of Hanoi algorithm: `https://www.ibm.com/developerworks/community/blogs/jfp/entry/towers_of_hanoi_at_large1`

[5]Image source: `https://upload.wikimedia.org/wikipedia/commons/0/07/Tower_of_Hanoi.jpeg`

Sources:

https://skerritt.blog/divide-and-conquer-algorithms/

https://www.transum.org/Maths/Investigation/Tower_Of_Hanoi/