**This lab is worth 5%. Each lab stream has been divided up into 50-minute slots, during one of which you will sit the practical test.**

*Please make sure you come to your scheduled lab stream, and don't be late.*

---

## Overview

Most of the labs that we get you to do during this course are formative in nature; i.e. you learn new things and acquire new skills during the course of completing them. This lab is a bit different. Instead of being formative this lab is summative in nature. It allows you (as well as us) to look back and see what knowledge and skills you have acquired so far.

In the previous assessed labs you could complete the work at any time before the deadline, using whatever resources you wished to. In this lab you must complete the task during the allocated time, with minimal resources at your disposal. Using only a terminal window, a basic text-editor, javac, and java, you must write some programs which meet the given specifications.

At first you might think this sounds a bit daunting; however we are *not* asking you to create your programs for the first time during your allocated slot. You should have already written all of the code required for this test during previous labs. All you need to do during this lab is to be able to reproduce your previous work.

We encourage you to write your code initially using whatever resources you need. Once you have done this, and are sure that your programs are working correctly, try to write them again without using any resources (you might need to peek occasionally). Keep doing this until you can write the code completely unaided. The best and recommended way to prepare for the test is to clearly understand what your code is doing. That way you don't have to write it exactly the same way each time. If you prepare well for this lab then you will find it pretty straightforward. Writing code like this, without any outside assistance, will build your confidence to tackle more demanding programming tasks.

---

## Resources

The skeleton code, **`Coins.java`** and **`Tower.java`**, from week 2 and week 3 will be provided, as will the associated handouts.

**Note: You are not permitted to access your home directory, or any other files or computers, nor may you use the Internet during this lab.**

## Part A (1%)

The skeleton code, **`Coins.java`**, includes some data fields and a basic constructor. To this, add two functions:

**`countHeads()`** A method that returns an **`int`** which is the number of occurrences of "heads" in the coin tosses.

**`toString()`** A method that returns a **`String`** representation of the coin tosses, using **`H`** to represent heads and **`T`** to represent tails.

## Part B (1%)

Part B requires that the **`toString()`** method from Part A has been completed. Add to your **`Coins`** class two more constructors:

**`Coins(String c)`** Creates a **`Coins`** object from a **`String`** consisting entirely of the characters **`H`** and **`T`** (i.e., the result of applying **`toString()`** to the constructed object should be the original string **`c`**).

**`Coins(int length)`** Constructs a **`Coins`** object consisting of a series of **`length`** coins – the value of each coin should be determined by a random coin toss.

Also add one more method:

**`countRuns()`** Returns an **`int`** which is the number of runs in this sequence of coins (a run is a block of coins all showing the same face, so for example in **`HHTHHHTTT`** there are four runs namely **`HH`**, **`T`**, **`HHH`**, and **`TTT`**.

## Part C (1%)

Create an application file called **`RecursiveApp.java`** that contains the following two functions:

**`digits(long n)`** Returns a **`long`** equal to the number of digits of its argument

**`sumOfDigits(long n)`** Returns a **`long`** equal to the sum of the digits of **`n`** modified by the sign of **`n`**. That is, **`sumOfDigits(257)`** should return 14, whereas **`sumOfDigits(-257)`** should return -14.

Your methods should be static and also use recursion.

### Part D (1%)

Add the following methods to the provided class **`Tower.java`**:

**`height()`** A method that returns an **`int`** equal to the height, i.e., number of blocks, in the tower.

**`count(char c)`** A method that returns an **`int`** equal to the number of blocks equal to **`c`** in the tower.

Both methods should be recursive.

### Marking

You must complete your programs and get them marked by a demonstrator before the end of your 50-minute time slot. All of your programs should be in the **`week05`** package. You can check that your programs are working using *241-check* as usual. You can run *241-check* (but not *241-submit*) in the lab beforehand, as well as during the test. **Note that comments are not required**. Also, no style checks are performed (although it's still a good idea to keep your code tidy). Each of the four parts of this test are worth 1%. If you complete them all correctly then you will get receive an extra 1%. This means that possible marks for this test are 0, 1, 2, 3, or 5. If you are unable to successfully complete all of the tasks within the allotted time you will be given the opportunity to do the test again later in the semester.