**G52ADS 2014-15: Coursework ONE.  
"Analysis of Scaling"**

**WEIGHT: This coursework will contribute 8% of the total module score.**  
**DEADLINE: 4pm, Wednesday Oct 15, 2014.**  
 **Description**

Firstly you should download the file Main.java from Moodle. Your first task should be to read it carefully, and then check that you can compile and run it. Then there are three parts to the C/W:

1. “**Counting**”. In the file “Main.java” you are given a fragment piece of code for a method “p”. It contains lines “c += ??” to increment a counter of primitive operations. You are required to modify these increments to do some approximate counting of the "primitive operations" that it performs. Using the template for the report, these should be reported in the table – along with brief explanation of the increments – especially in those cases that are less obvious.
2. “**Graph**”. Using the supplied code, modify the Main routine as appropriate, for example changing the range of values of the input size, and the number of runs at each size. Take the resulting outputs and use them to produce an initial scatterplot graph of how the behaviour. Then add to the graph extra ‘lines’ that illustrate the scaling of different aspects of the runtime. The two main aspects of runtime that you should consider are the worst and the best runtime. However, you may also consider the average runtime. The lines you add to the graph should support the analysis in part C.
3. “**Analysis**”. In the final part of the report give an analysis and interpretation of the code and runtimes. Issues that you should discuss include:
   1. A description of the code – what the function does, and the way it is initialised.
   2. Big oh and omega analyses and interpretations of
      1. The worst case runtime at each value of n
      2. The best case runtime at each value of n
      3. (harder, optional) The average case runtime at each value of n

The questions are deliberately a bit open-ended and under-specified. It is intended to mimic the case in which you are given some component of a program to analyse and are required to "say something useful about the highlights of the scaling behaviour". You do not need, and should not attempt, to produce a complete analysis.

This is the first coursework and will be marked generously. You primarily just need to demonstrate that you understand.

**Be aware that will be required to explain your submitted answers to the tutors during a lab session.**

**I am well aware that many people find big-Oh rather challenging, and so the coursework will be marked appropriately (generously).**

## Submission Requirements

You need to submit a report (both soft and hardcopy) and your source code (softcopy only).  
  
**SOFTCOPY:**  
By the deadline, you must submit in Moodle TWO files:

1. The electronic report. This should be the word template docx file.
2. Your own version of Main.java.

The total length of the report section (excluding copies of portions of your code) must be no more than TWO sides.

**HARDCOPY:   
  
You MUST also submit a hardcopy of the report in the standard fashion (time-stamped with coversheet into the coursework letterbox) by the deadline. Please print single sided – this makes it easier for markers to add annotations.**

Remark: The double submission system is to make sure that if one is somehow lost then the other will be present (which protects you). It also gives me the option to check your code electronically, both to see it works and to check for plagiarism. As usual, you should be regularly copying all your work to the Unix servers (the H drive) as these are backed up. If you get a mark of zero and claim that your submission was lost then I will be checking the Moodle system and (possibly) the backups, but if it is not found on Moodle or the University/School fileservers system then I will have no option but continue to treat is as not submitted.

## Marking Scheme

The division of marks available between the parts A, B and C is not rigid, but is roughly:

* 30% Part A
* 50% Parts B and C (counted together as they are closely linked).
* 20% Individual session. (On pass/fail basis) Attendance and participation in short (less than 5 minutes per person) individual meeting with tutors. Details will be provided later; but you should expect at least to be able to explain what you did and answer questions about your submission; and in return you will get **personal specific feedback**. The intention is that anyone attending and making a bona fide effort to participate will get a pass (and get the full 20%).

### Marking Criteria:

The main criteria:  
  
The effectiveness and reasonableness of your experimental studies and the associated analyses. The quality of analysis of the functions – ideally it should be both clear and brief.

**Late submissions allowed for up to a FIVE days after the deadline**, and are penalised at the standard University rate (5% per day, excluding weekend days). That is the latest time to submit is 4pm Monday 20th October, and this will incur a penalty of 15%.  
  
(The shorter than usual deadline is that that can release overall feedback in time for the lab sessions – otherwise have to wait until the last submissions.)

**Plagiarism Policy**

This coursework must be all your own work. You should remember that the coursework submissions will be archived, and plagiarism detection tools may be used at any time. Plagiarism is a very serious offence! Read the University regulations. If at all in doubt about whether something is allowed, then consult me or your personal tutor.

**Objective**

LEARNING OBJECTIVES OF THE C/W: The open-endedness is a deliberate part of the training that this C/W intends to give you. It is vital to develop the skill to decide for yourself what experiments to run, and exactly which graph(s) to produce, rather than just following a precise list produced by someone else. It is generally not possible to decide all experiments in advance, and so it needs to be done interactively and iteratively - if I specified the experiments then it would basically tell you the answers. This intends to help get start on the process of learning to design experiments. For example, in doing G52GRP if your program is computational intensive then you will need to design appropriate tests. If you are in industry and asked to understand the scaling of a complex piece of code then you cannot expect your boss to provide a list of the experiments to do; but will be expected to figure it out for yourself. Specifically, the objectives of this coursework are to:

* give some 'hands-on' experience with analysis of efficiency of algorithms
* develop an experience of how some common functions scale
* to reinforce the definitions of big-Oh (and family)
* to be able to match big-Oh techniques to plots of runtime versus input size, n.
* for the big-Oh family to become less abstract, and more linked with concrete results

**Hints and Suggestions**

I will maintain a help/FAQ file online

Last updated: 14-Oct-2014   
Contact: Andrew.Parkes 'at' nottingham.ac.uk