University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	**Advanced Systems Programming (M)**			
Coursework Number	**Summative exercise 1**			
Deadline	**Time:**	**4:30pm**	**Date:**	**17 February 2020**
% Contribution to final course mark	**10%**			
Solo or Group ✓	**Solo**	✓	**Group**	
Anticipated Hours	**10**			
Submission Instructions	**Submit via Moodle**			
Please Note: This Coursework cannot be Re-Assessed				

Code of Assessment Rules for Coursework Submission

Deadlines for the submission of coursework which is to be formally assessed will be published in course documentation, and work which is submitted later than the deadline will be subject to penalty as set out below.

The primary grade and secondary band awarded for coursework which is submitted after the published deadline will be calculated as follows:

(i) in respect of work submitted not more than five working days after the deadline
   a. the work will be assessed in the usual way;
   b. the primary grade and secondary band so determined will then be reduced by two secondary bands for each working day (or part of a working day) the work was submitted late.

(ii) work submitted more than five working days after the deadline will be awarded Grade H.

Penalties for late submission of coursework will not be imposed if good cause is established for the late submission. You should submit documents supporting good cause via MyCampus.

Penalty for non-adherence to Submission Instructions is 2 bands

You must complete an "Own Work" form via https://studentltc.dcs.gla.ac.uk/ for all coursework

Advanced Systems Programming (M) 2019-2020 – Exercise 1

Dr Colin Perkins, School of Computing Science, University of Glasgow

3 February 2020

Introduction

Programming languages and systems have traditionally used one of three approaches to managing memory. These are that the system either provides a tracing garbage collector, or it supports reference counting with automatic memory reclamation when the reference count reaches zero, or it relies on the programmer to manually allocate and free memory. There are advantages and disadvantages to each, but the fact that all three are widely used suggests that none of them are suitable for all problem domains.

A new automatic memory management scheme that is gaining attention is *region-based memory management*. This is used in the Rust programming language[1], and in the older Cyclone research language[2]. Region-based memory management tries to provide effective automatic memory management, without the indeterminism and overheads of a garbage collector, by tracking ownership of data and using this understanding of ownership to automatically deallocate objects when they go out of scope.

In this summative exercise, you will study region-based memory management, and discuss its advantages and problems when compared to other approaches to memory management.

Summative Exercise

There are three parts to this exercise. You should prepare and print a single report that includes your answers to all three parts.

In part one you should research and prepare a description of how memory management works in the Rust programming language, and how the Rust language manages ownership of data, including different pointer types and borrowed references to data. You should describe when memory allocations occur in Rust programs, how Rust tracks ownership and borrowing of data, and when it deallocates memory. *[10 marks]*

In part two, you should compare and contrast how memory management work in Rust with memory management in the C programming language. Discuss how these two languages compare in terms of 1) programmer effort to manage memory; 2) efficiency; and 3) safety and flexibility. *[10 marks]*

In part three, you should compare and contrast region-based memory management using ownership tracking, as used by Rust, with memory management in garbage collected languages. You should highlight the relative advantages and disadvantages of the approach used in Rust as compared to a garbage collected language. With the aid of sample code fragments, discuss the types of program that Rust makes easy to write, and the types of program that are difficult, or impossible, to write in Rust. Explain what language design decisions make such programs easy or difficult to write. *[10 marks]*

[1] `https://rust-lang.org/`

[2] `http://cyclone.thelanguage.org`

Submission

You should submit a single printed report covering three of the topics described above. A mark out of 30 will be assigned to each submission, weighted as noted earlier. This mark will be converted to a percentage, then used to assign a band on the University's 22 point scale.

There is no single correct answer for this exercise. Answers can legitimately argue for or against the suitability of Rust, or about the appropriateness of the features discussed. Marks will be assigned for technically correct descriptions of the various language features, for properly researched arguments that have well-reasoned justification, and for presenting clear and coherent arguments.

Print your report on A4 paper, formatted in two columns, using the Times Roman font in 10pt, with 1.5cm margins (i.e., using a format that matches this page of the handout). If you use LaTeX to prepare your document, the following preamble will format your submission appropriately:

```
\documentclass[10pt,a4paper,twocolumn]{article}
\usepackage[cm]{fullpage}
\usepackage{newtxtext}
\usepackage{newtxmath}
\begin{document}
...
```

You are not required to use LaTeX. Your report must not exceed six pages in length, including all figures, tables, code samples, and any references. Length is not an indication of merit: if you can cover the required material in less than six pages, then please do so.

You must submit your report before 4:30pm on 17 February 2020. Following the code of assessment, late submissions will be accepted for up to 5 working days beyond this due date. Late submissions will receive a two band penalty for each working day, or part thereof, the submission is late. Submissions that are received more than five working days after the due date will be awarded a band of H.

Submissions must be made via Moodle. This problem set is worth 10% of the mark for this course. Submit a single PDF file entitled `asp-ex1-`*`GUID`*`.pdf`, replacing *`GUID`* with your GUID (your student number followed by the first letter of your surname). Submissions that do not follow these submission instructions will be given a two band penalty. Penalties will be strictly enforced.