COM00005I

BSc, BEng, MEng, MMath Examinations, 2011–2012

All Undergraduate Computer Science Programmes
Stage Two

Principles of Programming Languages (POPL)

Open Examination

Issued at:
Noon: Aut/9/Wed (7 December 2011)

Submission due:
Noon: Aut/10/Wed (14 December 2011)

Your attention is drawn to the Guidelines on Mutual Assistance and Collaboration in the Students' Handbook and the Department's 'Guide to Assessment Policies and Procedures' (http://www.cs.york.ac.uk/exams/statementonassessment/).

All queries on this assessment should be addressed to
Alan Wood (Part 1) `wood@cs.york.ac.uk`**, or**
Alan Burns (Part 2), `burns@cs.york.ac.uk`.

There are two parts in this assessment. You must complete both parts, which carry equal marks. Each part is divided into tasks, and the relative weight for each task is indicated as a percentage of the total marks for the assessment. All tasks must be attempted.

Submission

- All submissions should be done *electronically*, using the Department's online submission system.[1] *Hardcopy submissions will not be accepted.*
- All submitted material must be in PDF format.[2]
- Separate documents should be submitted for each part.
- Each item submitted should indicate clearly for which task it is the solution.

[1] `https://www.cs.york.ac.uk/submit/index.php`
[2] PDF output from a Word document is available by using the Departmental Word installations. PDF output from a LaTeX document is obtained by using `pdflatex`.

Part I.

There is one task in this part:

Task 1 Write an essay, of approximately 2500 words[3] (not including program code), with the following specification: 50%

Title:

A Review of Static and Dynamic Scoping in Programming Languages.

Abstract All programming languages allow *names* to to be associated with *values* by means of *definitions*, and a name is said to be in the 'scope' of its definition. When a name is mentioned in a program, its definition (if any) must be known, in order for its 'invocation' to make sense. However, most languages allow names to be re-defined in a program — the rules for determining to *which* definition a name refers, are called the *scoping* rules.

There are two principal scoping methods: *static* and *dynamic*. This essay will firstly explain these two concepts, comparing their similarities and differences. The uses of the two methods will be discussed, illustrating these with code from actual programming languages. Finally the essay will conclude with a critical summary of the pros and cons of static and dynamic scoping for present day programming problems.

Notes

1. Start your essay with the *exact* title and abstract as given above.
2. Use the principles of writing that were taught in the Stage 1 SKIL module.
3. Full references, properly formatted, to sources of material *must* be given.[4]
4. No more than half of the items in your bibliography can be to URLs alone — the others must be to 'printed' sources.
5. The textbook by Sebesta gives a good introduction to the topic, so you might start from there. However, you *must* draw on other, fully-referenced, sources.

[3]About 5 typed (10pt), single-spaced, A4 pages

[4]Remember, we can Google as easily as you!

6. You *must* include appropriate program illustrations from at least *three* different programming languages, and these must all be syntactically correct. These illustrations should *not* be counted as part of the 2500 words of the essay.

7. In order to pitch the essay at the correct level, you should assume the reader has the technical knowledge of a typical CS student just starting Stage 2, who has not yet attended the POPL module.

Marking

Points for which marks will be given include:

- Clarity of explanation.
- Quality and choice of references.
- Appropriate program examples.
- Quality of argument
- Clarity of essay structure.
- Quality of the conclusion.

Part II.

There are three tasks in this part:

Task 2 Write a short review, of approximately 500 words (at most one typed (10pt) A4 page), with the following specification: 15%

The cigarette smokers problem is a well known concurrency problem, originally described in 1971 by S. S. Patil. Assume there are three compulsive smokers around a table, each of whom has an infinite supply of one of the three necessary ingredients - one smoker has an infinite supply of tobacco, another has an infinite supply of paper, and the third has an infinite supply of matches. Assume there is also a non-smoking agent with an infinite supply of all three ingredients. The agent selects two of the ingredients (non-deterministically), and places them on the table. This allows one of the smokers to obtain their supply and smoke a cigarette for a while. Meanwhile, the agent, seeing the table empty, again chooses two ingredients at random and places them on the table. This process continues forever.

Using sources available on the internet consider how this problem has been discussed and solved since 1971.

Task 3 Considering the different ways of supporting communication and synchronisation in concurrent programming languages what facilities would you use to program the cigarette smokers problem. Give reasons for your choice or choices. 10%

Task 4 A variation on the cigarette smokers problem is that there are three agents each producing two of the ingredients (i.e. one agent produces tobacco and paper, another produces paper and matches, and the other produces matches and tobacco). Develop a program that implements this system of 3 smokers and 3 agents. Your program should allow smokers to smoke concurrently if there are resources available, but there is limited space on the table, and hence agents must be prevented from generating excessive ingredients. At most there should be only one of each ingredient free on the table at any time. An attempt should be made to be fair to each smoker in terms of their access to the ingredients.

Your program should represent each smoker and agent as a task/thread and use a monitor or monitors to control access to the resources (ie. the ingredients). An execution of your program should simulate the behaviour of the smokers and agents over a number of iterations. To simulate the act of smoking a delay/sleep routine should be called.

You can program this is either Ada, Java or Pascal-FC. A description of your code should also be provided, including any further assumptions you make about the problem. Example runs of your program should be provided. $\boxed{25\%}$

Note. The program must implement the above variant of the cigarette smoker's problem, not the original problem. Only one language solution is required. If more than one program is submitted only the first program will be marked.

POPL Open Assessment
Marking Notes

Part I

Task 1

Expect the following topics:

- Definition of scope (in general)
- Distinction between static and dynamic (in general, and wrt scope in particular).
- Languages which have static, dynamic, or both
- Syntactical variants for flagging scope type.
- Comparison in language(s) which have both types — how does it differ.
- Implementation issues.
- Discussion of relative merits.

Expect *proper* structure — Intro, Background, Examples, Conclusion, Refs.
Allow 1500–3000 words with no loss of marks. Outside that range loss of marks if quality doesn't merit the extra length (or conciseness!)

Part II

Task 2

There is a lot on information on the web about this problem. Expect some historical information, also an overview of the solutions that seem to be discussed - these seem to be mainly semaphore based. A comparison of some of the published solutions would lead towards full marks. This could include comparison between different language features (semaphores, monitors, message passing) and different languages.

Task 3

Here we are looking for a comparison between the three main language models that have been dealt with in the course: semaphores, monitors, message passing.
There are some straightforward synchronisation issues (eg tobacco cannot be used until it is produced) that can be catered for my simple primitives such as semaphores (as most of the published solutions do).

The resources (tobacco, paper and matches) could however be contained within monitors that would protect their use. There could be one for all resources, or one each for each resource. Procedures and consumers would interact via the monitors.

Finally the agent(s) and the three smokers could interact directly to pass the resources around.

Some discussions or deadlock and fairness would be needed for high marks.

Task 4

This variant with three agents is not available on the web (as far as we were able to ascertain). The use of three agents does encourage the use of a single monitor.

Some level of freedom has been given to choose any one of the three languages covered on the course. No credit will be given to the choice made - ie all are equivalent.

The single monitor solution will need 6 methods to call. The three languages deal with synchronisation differently, so the details will differ. It will be necessary to argue that there is some level of fairness and deadlock free execution in the solution provided.