Reference language for Higher Computing Science Question Papers

Note: This document replaces the document "Pseudocode for Higher Computing Science Question Papers", published in June 2013.

1 Overview

The ability to reason about code is increasingly being seen as a crucial part of learning to program. For example, if you can't explain in precise detail what a fragment of code does, you can't debug. If you can't explain the code you've just written to someone else, how can you justify any of the decisions you made in creating it, and then demonstrate any level of understanding?

To assess candidates' ability to reason about programs, programs must be presented in assessment questions. This document contains a specification for a reference language designed for setting such questions, developed in collaboration with Prof. Greg Michaelson of Heriot Watt University, Dr. Quintin Cutts of the University of Glasgow, and Prof. Richard Connor of Strathclyde University. It is suitable for use in schools and FE/HE institutions. It enables assessors, teachers and candidates to work to one well-defined notation.

Note: in earlier versions of this document, this reference language was referred to as "pseudocode". Given that the language presented here is formally-defined, and pseudocode is not, the term "reference language" is preferred. A formally-defined language is required because it is a candidate's ability to understand and analyse code in such languages (including programming languages) that should be assessed.

The use of a reference language supports SQA's decision to allow centres to use the programming language of their choice for teaching and learning, as long as assessors ensure that candidates have mapped their understanding from the language of instruction across to the reference language. This focus on concepts that are shared among programming languages is potentially a major lever in deepening understanding of computation in general.

Although the idea of a clearly-defined reference language may seem daunting, it is not in fact so different from the "pseudocode" that has been used for years in SQA question papers. It has simply been regularised, so that current and new teachers, assessors, question paper-setters and candidates will all be working to the same definition.

This document presents a reduced specification suitable for the SQA Higher Course. Documents providing the full specification, and giving more extensive examples of use, will be available in due course.

In reviewing this specification, bear in mind its primary purpose:

- ♦ Where candidates may be instructed using one of a range of languages, this clearly-defined reference language should enable code **to be presented** to candidates under closed assessment conditions such that they can reason about it.
- ♦ Candidates are **not** expected to **write** code in the reference language, given that assessors should be able to mark solutions written in a range of languages commonly used for teaching — and so candidates can use the language of their choice.

Note that assessors and candidates may choose to use this reference language as a tool to support program design, but this is **not its primary purpose**. The elision feature <*…*> presented in Section 6 enables the inclusion of steps that have not been fully worked out yet.

The aim in the rest of the document is to present the reference language principally via a small number of examples. In reading through the examples and specification, attachment to particular constructs, or to 'my favourite construct in language X', should be avoided — it is the **concepts** that are the major focus.

2 Introducing the reference language by example

The following typical programming examples show that solutions in our specified reference language do not differ markedly from those in any pseudocode notation.

The first example is for the problem:

> *Read in a number representing a temperature in degrees Celsius and write it out as a value in degrees Fahrenheit. If the Celsius value is C, then the Fahrenheit value, F, is calculated as follows: F = (9 / 5) * C + 32.*

Using our reference language, this would be written as follows:

```
RECEIVE c FROM KEYBOARD
SET f TO ( 9.0 / 5.0 ) * c + 32
SEND f TO DISPLAY
```

An immediate observation is that the keywords are written in CAPITALS. In any representation of programming language code, it is useful for the reader to distinguish easily between the language's keywords and other names created by the user. Using bold and underline is one technique. Capitalisation is another, and it has been chosen to facilitate keyword highlighting when writing examples on paper or on the board, where emboldening is not practical, and underlining can get messy.