S

School of Computing, Engineering and Mathematics

Assessment Brief Form

Module Title:	Programming Languages, Concurrency and Client/Server Computing
Module Code:	CI346
Author(s)/Marker(s) of Assignment	Jim Burton & Aidan Delaney

Assignment No:	One
Assignment Title:	Programming Language comparison
Percentage contribution to module mark:	20%
Weighting of component assessments within this assignment:	N/A
Module Learning Outcome/s Covered: (Refer to module syllabus)	LO1: Critically comment on and be able to compare essential features taken from a range of programming languages covering several paradigms. LO2: Demonstrate an understanding of the relationship between programming languages and machine and software architectures.

Assignment Brief and Assessment Criteria:
A comparison of **either** Java or C++ with Haskell (that is, you should either compare Java with Haskell or C++ with Haskell). This comparison should be illustrated with some well chosen **sample code** to show the differences and similarities between the languages. Remember to include in your comparison the different or similar ways in which issues such as **modularity**, **Input/Output**, **code reuse** and **concurrency** are handled. Examples over a few lines should form part of an appendix. The expected length of the comparison is 4-5 pages, excluding code examples in the appendix. **What is expected** An overview of the major differences between the programming languages (Java or C++) and Haskell. Remember, this should be at a *high level* and should concentrate on

the *conceptual approaches* of the languages, rather than details of their implementation or tooling
environment. For example, if you try to port a non-trivial program written in Java or C++ into Haskell, a direct translation will not be possible due to widely differing approaches to problem-solving. This is true despite the fact that all three languages are general purpose and Turing complete.

As part of this overview you should consider how typical issues that arise in software development may be handled. For example, you might examine the following issues:

- What features are provided to enable *modularity* and *separation of concerns*?
- What features are provided to enable *code reuse* (e.g. what types of polymorphism are available in each language)?
- How do the approaches to *Input/Output* and *functional purity* differ?
- How does each language allow the programmer to prevent the *corruption of shared data* between concurrently executing threads/ tasks?
- Are there particular domains in which each language could be considered *more fit-for-purpose* than the other? Justify your opinions on this.

You should also include other questions and comparisons not in the above list. Most importantly, you should include a **reflection on what you believe the implications of your findings to be**.

You will be penalised if you go above the suggested page count. The aim is to write succinctly and to concentrate on the essence of the features discussed.

Assessment Criteria

Standard GEAR and CEM marking criteria will be applied. Specifically, the following criteria will be considered:

- Depth and quality of the analysis.
- Illustrative code samples provided.
- References supplied in a recognised academic format.
- Absence of typographic and grammatical errors and the use of an appropriate academic tone of writing.

Date of issue:	01/10/2016
Deadline for submission:	12/01/2017
Method of submission:	Submission is via studentcentral using Turnitin.
Date feedback will be provided	09/02/2017

1. *A copy of your coursework submission may be made as part of the University of Brighton's and School of Computing, Engineering & Mathematics procedures which aim to monitor and improve quality of teaching. You should refer to your student handbook for details.*

2. *All work submitted must be your own (or your team's for an assignment which has been specified as a group submission) and all sources which do not fall into that category must be correctly attributed. The markers may submit the whole set of submissions to the JISC Plagiarism Detection Service.*