1. KentVision Code and title of the module

COMP6630 – Programming Languages: Applications and Design

1. **School or partner institution which will be responsible for management of the module**

School of Computing

1. **The level of the module (Level 4, Level 5, Level 6 or Level 7)**

Level 6

1. **The number of credits and the ECTS value which the module represents**

15 credits (7.5 ECTS)

1. **Which term(s) the module is to be taught in (or other teaching pattern)**

Autumn or Spring

1. **Prerequisite and co-requisite modules**

Pre-requisites:
COMP5450 - Functional Programming;
COMP5180 - Algorithms, Correctness and Efficiency

1. **The programmes of study to which the module contributes**

Computer Science and related programmes, including year in industry versions.

1. **The intended subject specific learning outcomes.
On successfully completing the module students will be able to:**
	1. Describe the behaviour of programs written in a small imperative and functional programming language using operational semantics
	2. Translate these ideas from theory to practice by implementing an interpreter for the language
	3. Adding basic features to a base language.
	4. Writing algorithms using imperative (while loops) and functional (recursive operators) languages.
2. **The intended generic learning outcomes.
On successfully completing the module students will be able to:**
	1. Demonstrate comprehension of the trade-offs involved in design-choices.
	2. Make effective use of IT facilities for solving problems.
	3. Manage their own learning and development, through self-directed study and working on continuous assessment.
3. **A synopsis of the curriculum**

This module shows students what trade-offs are involved in designing a programming language, and how those trade-offs ultimately influence programmer productivity. The module starts with a quick, example-based introduction to the basics of programming languages. It then continues with a series of problems involved in the design of several programming languages.

As an example, the students will be shown the design difference between imperative languages with state and functional programming languages with types and general recursion. These will form the basis to construct more involved programming languages and realistic, e.g. non-determinism, polymorphism, effects and exceptions.

Also, they will be taught how to translate the mathematical description of a programming language into its corresponding implementation.

Indicative examples are:

* Implementation of interpreters for C-like and Haskell-like languages.
* Problem solving. The problems will involve concepts such as parsing, evaluation, trees, graphs, memoization, randomization.
* Adding Language features: first order functions, polymorphism, effects, exceptions, types, algebraic data types.
1. **Reading list (Indicative list, current at time of publication. Reading lists will be published annually)**

Winskel, G. (1993) The Formal Semantics of Programming Languages: An Introduction. MIT Press

Bird, R., Gibbons J. (2020). Algorithm Design with Haskell. Cambridge University Press

Bird, R. (2014). *Thinking Functionally with Haskell*, Cambridge University Press.
Hutton, G. (2016) *Programming in Haskell*, 2nd edition. Cambridge University Press
Krishnamurthi, S. (2015) [available for free online]. *Programming Languages: Application and Interpretation*.

Streicher, T. (2006) Domain-Theoretic Foundations of Functional Programming. World Scientific Publishing Company

1. **Learning and teaching methods**

Total contact hours: 32 hours

Private study hours: 118 hours

Total study hours: 150 hours

1. **Assessment methods**
	1. Main assessment methods

Programming Assignment A1 (30%)

Programming assignment A2 (30%)

Class Exercises (10%)

Examination 2 hours (30%)

13.2 Reassessment methods

Like for like.

1. ***Map of module learning outcomes (sections 8 & 9) to learning and teaching methods (section12) and methods of assessment (section 13)***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Module learning outcome** | 8.1 | 8.2 | 8.3 | 8.4 | 9.1 | 9.2 | 9.3 |
| **Learning/ teaching method** |  |  |  |  |  |  |  |
| **Private Study** | **X** | **X** | **X** | **X** | **X** | **X** | **X** |
| **Lectures** | **X** | **X** | **X** |  |  |  |  |
| **Practical Sessions** | **X** | **X** | **X** | **X** | **X** | **X** |  |
| **Assessment method** |  |  |  |  |  |  |  |
| Programming assessment |  | **X** | **X** | **X** |  | **X** |  |
| Participatory exercises | **X** | **X** | **X** | **X** |  | **X** | **X** |
| Paper presentation | **X** |  |  |  | **X** | **X** | **X** |
| Examination | **X** |  |  | **X** | **X** |  |  |

1. **Inclusive module design**

The School recognises and has embedded the expectations of current equality legislation, by ensuring that the module is as accessible as possible by design. Additional alternative arrangements for students with Inclusive Learning Plans (ILPs)/declared disabilities will be made on an individual basis, in consultation with the relevant policies and support services.

The inclusive practices in the guidance (see Annex B Appendix A) have been considered in order to support all students in the following areas:

a) Accessible resources and curriculum

b) Learning, teaching and assessment methods

1. **Campus(es) or centre(s) where module will be delivered**

Canterbury

1. **Internationalisation**

The topics addressed by this module relate to a field which is of international importance, given the global role of computer programming in today's technological innovation. The programming languages covered by this module are international, being identical worldwide and independent of traditional spoken language.

**FACULTIES SUPPORT OFFICE USE ONLY**

**Revision record – all revisions must be recorded in the grid and full details of the change retained in the appropriate committee records.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date approved | Major/minor revision | Start date of the delivery of revised version | Section revised | Impacts PLOs (Q6&7 cover sheet) |
| 15/12/2022 | Major | September 2023 | 6,8,9,10,11 | No |
| Aug 2023 | Minor | September 2023 | 13 | No |