Confirmation that this version of the module specification has been approved by the School Learning and Teaching Committee:

19 December 2014………………………………………………….(date)

**MODULE SPECIFICATION**

1. **Title of the module**

CO658 Programming Language Implementation

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

School of Computing

1. **Start date of the module**

Academic year 2015/2016

1. **The number of students expected to take the module**

40

1. **Modules to be withdrawn on the introduction of this proposed module and consultation with other relevant Schools and Faculties regarding the withdrawal**

None

1. **The level of the module (e.g. Certificate [C], Intermediate [I], Honours [H] or Postgraduate [M])**

H

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

15 (7.5 ECTS)

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

Either Autumn or Spring term

1. **Prerequisite modules**

CO545 Functional and concurrent programming

CO527 Operating systems and architecture

CO518 Algorithms, correctness, and efficiency

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

Computer Science and related programmes

1. **The intended subject specific learning outcomes**

Students who complete this module will:

* 1. understand how a computer program in a high-level, imperative language is translated into machine code;
	2. understand how a program is executed, including run-time system support;
	3. understand a variety of techniques that a compiler uses to improve the efficiency of its generated code;
	4. understand how to represent programs as data in a typed functional language;
	5. be able to implement basic compiler optimisation techniques;
	6. be able to evaluate a program’s performance; and
	7. be able to work with and modify an existing code base.
1. **The intended generic learning outcomes**

Students who complete this module will:

* 1. Be able to clearly communicate the results of performance experiments;
	2. Be able to manage their own learning and development, through self-directed study and working on continuous assessment.
1. **A synopsis of the curriculum**

A study of techniques for interpreting and compiling programming languages, implementing them in a typed functional programming language (e.g., OCaml, Haskell). The module will outline a whole compiler from source to machine code, but will focus in depth on key algorithms and techniques. Possible in-depth topics include:

* writing interpreters,
* Hindley-Milner type inference,
* register allocation,
* garbage collection,
* abstract interpretation,
* static single assignment form.

The implemented language will be based on a simple imperative (e.g., Pascal-like) language with some extensions to address advanced topics in data layout (e.g., closures, objects, pattern matching). The course will be organized around a simple, but complete, example compiler that the student will have to understand and modify.

1. **Indicative Reading List**

Andrew W. Appel, *Modern compiler implementation in ML*, Cambridge University Press, 2004

Keith D. Cooper and Linda Torczan, *Engineering a compiler,* Morgan Kaufmann, 2011

Yaron Minsky, Anil Madhavapeddy, and Jason Hickey, *Real world OCaml,* O'Reilly Media, 2013

Alfred V. Aho, Monica S. Lam, Ravi Sethi and Jeffrey D. Ullman, *Compilers: Principles, Techniques, and Tools 2nd ed.,* Prentice Hall, 2007

1. **Learning and Teaching Methods, including the nature and number of contact hours and the total study hours which will be expected of students, and how these relate to achievement of the intended module learning outcomes**

33 hours of lectures (11.1, 11.2, 11.3, 11.4, 11.6, 11.7)

2 hours of terminal sessions, early in the term (11.4, 11.5)

50 hours of study and revision (11.1, 11.2, 11.3, 11.4, 12.2)

65 hours on assessed coursework (11.2, 11.3, 11.4, 11.5, 11.6, 11.7, 12.1, 12.2)

150 hours total

1. **Assessment methods and how these relate to testing achievement of the intended module learning outcomes**

Assessment will be through:

* 1 programming exercise (5%) testing 11.4;
* 2 practical assessments (55% total) testing outcomes 11.2, 11.3, 11.4, 11.5, 11.6, 11.7, 12.1, and 12.2;
* A 2 hour examination (40%) testing outcomes 11.1, 11.2, 11.3, and 11.4.

The programming exercise will be set early in the term to allow students to practice programming in the typed functional language to be used in the projects.

For one practical assessment, the students will (either individually or in groups) implement a more advanced algorithm of their choice for one part of the module’s simple compiler, and evaluate its impact on performance. Each student will write a short report explaining the measured performance benefits of their implementation, and explain their own contribution to the implementation in a short (e.g., 7 minute) viva. Students will be marked individually.

For the other practical assessment, the students will (either individually or in groups) implement an extension of their choice to the module’s simple compiler. Each student will write a short report explaining what the language extension does, and how the compiler extension implements it. They will also explain their own contribution to the implementation in a short viva. Students will be marked individually.

1. **Implications for learning resources, including staff, library, IT and space**

The module will require rooms for terminal sessions during the first 2 or 3 weeks of term. The terminal sessions can be staffed by the core lecturing team and selected sessional teachers.

1. **The School recognises and has embedded the expectations of current disability equality legislation, and supports students with a declared disability or special educational need in its teaching. Within this module we will make reasonable adjustments wherever necessary, including additional or substitute materials, teaching modes or assessment methods for students who have declared and discussed their learning support needs. Arrangements for students with declared disabilities will be made on an individual basis, in consultation with the University’s disability/dyslexia support service, and specialist support will be provided where needed.**
2. **Campus(es) or Centre(s) where module will be delivered:**

Canterbury