UNIVERSITY OF KENT – CODE OF PRACTICE FOR QUALITY ASSURANCE

MODULE SPECIFICATION

1 **The title of the module: CO326** Functional Programming

2 **The Department which will be responsible for management of the module**: Computer Science

3 **The start date of the module:** September 2004

4 **The cohort of students (onwards) to which the module will be applicable.** 2009/10

5 **The number of students expected to take the module:** 170

6 **Modules to be withdrawn on the introduction of this proposed module and consultation with other relevant Departments and Faculties regarding the withdrawal:**

CO310 Foundations of Computer Science is withdrawn. The Mathematics department has been consulted.

7 **The level of the module** (eg Certificate [C], Intermediate [I], Honours [H] or Postgraduate [M]): C

8 **The number of credits which the module represents:** 15

9 **Which term(s) the module is to be taught in (or other teaching pattern):** Spring

10 **Prerequisite and co-requisite modules: Pre-requisite:** CO322, Co-requisite: CO325

11 **The programmes of study to which the module contributes:** BSc Mathematics with Computer Science, including Year in Industry variants.

12 **The intended subject specific learning outcomes and, as appropriate, their relationship to programme learning outcomes:**

It is expected that upon completion of the module students should

- Have basic understanding of the concepts of Functional Programming: how evaluation operates, side-effect-free programming, the role of types, and be able to define and use specific types.
- Be able to use a functional programming language to write programs; be able to use recursion.
- Have basic understanding of Propositional and Predicate Logic: their syntax (connectives, quantifiers) and their semantics (truth tables, logical equivalences).
- Be able to write and evaluate expressions in Propositional and Predicate Logic.

13 **The intended generic learning outcomes and, as appropriate, their relationship to programme learning outcomes:**

- Make appropriate choices when faced with trade-offs in alternative designs. [B1]
- Deploy appropriate theory and practices in their use of methods and tools. [B5]

14 **A synopsis of the curriculum:**

Expressions, values and types. Introduction to the Hugs system (sessions and scripts). Numbers, booleans and characters. Function definitions, case analysis (guards). Approaches to testing programs. Polymorphic types. Lists and common list processing functions. Tuples. Higher order functions and currying. List comprehensions. Pattern matching, recursive function definitions. Library functions. Algebraic data types.

Propositional Logic: syntax, abstract syntax, truth tables. Predicate Logic: quantifiers, scope and renaming. Equivalences in either logic, e.g. de Morgan rules.

15 **Indicative Reading List:**

Simon Thompson, The Craft of Functional Programming

Paul Hudak, The Haskell School of Expression

J.K. Truss, Discrete Mathematics for Computer Scientists

16 **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 learning outcomes:**

150 study hours of which:

60 contact hours: 20 lectures,20 seminar classes, 20 terminal classes; of these, 14 are about Functional Programming, 6 are about Logic. The seminar classes directly following the Logic lectures are also about the subject of Logic, all other classes deal with Functional Programming.

Some classes will directly reinforce the lecture material, others indirectly by assisting students in their coursework.

30 hours private study in parallel with taught material

30 hours spent on exercises and assessments

30 hours pre-exam revision

Overall hours of study: 150.

17 **Assessment methods and how these relate to testing achievement of the intended learning outcomes:**

Both coursework and examinations assess all the learning outcomes, but in different environments that are either more controlled (examinations) or more conducive to program development (coursework).
Coursework (50%):

Assessments will address understanding and application of the concepts exhibited in the lectures to date, and put particular emphasis on the development of programming skills.

Examination (50%):

The examinations test all learning outcomes, with the proviso that programming skills are only tested on paper.

17 **Implications for learning resources, including staff, library, IT and space:**
Based on the assumption of 170 students there should be 11 student groups (class sizes no bigger than 16). Therefore: the module requires, for staff: class supervisors for 440 supervision hours; for space: 220 hours of terminal room space, 220 hours of seminar room space.

18 **A statement confirming that, as far as can be reasonably anticipated, the curriculum, learning and teaching methods and forms of assessment do not present any non-justifiable disadvantage to students with disabilities:**
We confirm, as far as can reasonably be anticipated, that the curriculum, learning and teaching methods only present justifiable disadvantages to students with disabilities.

Statement by the Director of Learning and Teaching: "I confirm I have been consulted on the above module proposal and have given advice on the correct procedures and required content of module proposals"

....
Director of Learning and Teaching Date

Statement by the Head of Department: "I confirm that the Department has approved the introduction of the module and will be responsible for its resourcing"

....
Head of Department Date

31 October 2006
August 2009