1. **Title of the module**

COMP5540 (CO554) Computing Theory and Concurrent Programming

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

Computing

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

Level 5

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)**

Spring term

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

CO320 (Introduction to Object-Oriented Programming) and CO322 (Foundations of Computing I)

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

BSc Computer Science for Health

BSc Computer Science for Health with a Year in Industry

1. **The intended subject specific learning outcomes.
On successfully completing the module students will be able to:**
	1. Understand specifications in formal logical notation
	2. Understand the expressiveness of various language formalisms
	3. Understand the difference between decidable and undecidable problems
	4. Have a basic understanding of the concepts of concurrent programming: software processes/threads, communication and synchronisation
	5. Be able to use effectively concurrent programming paradigms to design systems
	6. Be able to implement concurrent programs to solve specific problems using appropriate programming paradigms
	7. Have an understanding of deadlock, livelock, and starvation when designing concurrent systems
2. **The intended generic learning outcomes.
On successfully completing the module students will be able to:**
	1. Understand formal notation of various forms
	2. Understand trade-offs in alternative designs and make appropriate choices when faced with these
	3. Make effective use of IT facilities
	4. Manage their time and resources effectively
3. **A synopsis of the curriculum**
* Propositional & Predicate Logic, including proofs
* Formal languages: finite automata and regular expressions
* Elements of CFGs, Turing machines and decidability
* Introduction to Concurrency and Parallelism, threading model
* Synchronisation primitives for mutual exclusions & condition synchronization (e.g. Semaphores, Critical regions, Monitors)
* Safety and Liveness properties (e.g. Deadlocks, Livelocks, Starvation, Priority Inversion)
* Concurrency patterns (e.g., Producer-Consumer, Thread Pools, Lambda & Streams, Futures)
* Overview of other concurrency models (e.g. Message Passing, Tuple Spaces, CSPs, Actors, Transactions, Events)
1. **Reading list (Indicative list, current at time of publication. Reading lists will be published annually)**
* John Martin, Introduction to Languages and the Theory of Computation, McGraw-Hill, 4th Edition, 2010.
* George S. Boolos et al, Computability and Logic, Cambridge Press, 5th Edition, 2007.
* Brian Goetz et al, Java Concurrency in Practice, Addison-Wesley, 2015.

1. **Learning and teaching methods**

Total contact hours: 33

Private study hours: 117

Total study hours: 150

1. **Assessment methods**

13.1 This module will be assessed by 50% examination and 50% coursework.

2 hour unseen written examination (50%)

2 pieces of Coursework (25% each)

13.2 **Reassessment Method**

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* | *8.5* | *8.6* | *8.7* | *9.1* | *9.2* | *9.3* | *9.4* |
| **Learning/ teaching method** | **Hours allocated** |  |  |  |  |  |  |  |  |  |  |  |
| Lectures | 22 | **x** | **x** | **x** | **x** | **x** | **x** | **x** | **x** | **x** |  |  |
| Practical classes | 11 | **x** | **x** |  | **x** | **x** | **x** | **x** | **x** | **x** | **x** |  |
| Private study | 117 | **x** | **x** | **x** | **x** | **x** | **x** | **x** | **x** | **x** |  | **x** |
| **Assessment method** |  |  |  |  |  |  |  |  |  |  |  |  |
| *Coursework practical exercises* |  | **x** | **x** |  | **x** | **x** | **x** | **x** | **x** | **x** | **x** | **x** |
| *In-course test* |  | **x** | **x** | **x** | **x** |  |  | **x** | **x** | **x** |  |  |
| *Examination*  |  | **x** | **x** | **x** | **x** |  |  | **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**

Medway

1. **Internationalisation**

The topics addressed by this module relate to a field which is of international importance, given the global role of computer science in today's technological innovation. The theoretical techniques and methodologies 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) |
|  |  |  |  |  |
|  |  |  |  |  |