Board of Studies

Course Proposal Template

PROPOSED COURSE TITLE: Informatics 1 – Introduction to Computation

PROPOSER(S): Don Sannella

DATE: 19 Jan 2018, updated 21 Mar 2018

SUMMARY

This template contains the following sections, which should be prepared roughly in the order in which they appear (to avoid spending too much time on preparation of proposals that are unlikely to be approved):

1. Case for Support
– To be supplied by the proposer and shown to the BoS Academic Secretary prior to preparation of an in-depth course description

1a. Overall contribution to teaching portfolio

1b. Target audience and expected demand

1c. Relation to existing curriculum

1d. Resources

2. Course descriptor
- This is the official course documentation that will be published if the course is approved, ITO and the BoS Academic Secretary can assist in its preparation

3. Course materials
- These should be prepared once the Board meeting at which the proposal will be discussed has been specified

3a. Sample exam question

3b. Sample coursework specification

3c. Sample tutorial/lab sheet question

3d. Any other relevant materials

4. Course management

- This information can be compiled in parallel to the elicitation of comments for section 5.

4a. Course information and publicity

4b. Feedback

4c. Management of teaching delivery

5. Comments

- To be collected by the proposer in good time before the actual BoS meeting and included as received

5a. Year Organiser Comments

5b. Degree Programme Co-Ordinators

5c. BoS Academic Secretary

[Guidance in square brackets below each item. Please also refer to the guidance for new course proposals at http://www.inf.ed.ac.uk/student-services/committees/board-of-studies/course-proposal-guidelines. Examples of previous course proposal submissions are available on the past meetings page http://web.inf.ed.ac.uk/infweb/admin/committees/bos/meetings-directory.]

SECTION 1 – CASE FOR SUPPORT

[This section should summarise why the new course is needed, how it fits with the existing course portfolio, the curricula of our Degree Programmes, and delivery of teaching for the different years it would affect.]

1a. Overall contribution to teaching portfolio

[Explain what motivates the course proposal, e.g. an emergent or maturing research area, a previous course having become outdated or inappropriate in other ways, novel research activity or newly acquired expertise in the School, offerings of our competitors.]

This proposal is the outcome of discussions in the Curriculum Review Committee. The motivation is to rationalise Informatics teaching in the first semester of first year, combining the two existing 10-point courses (Inf1-FP and Inf1-CL) into a single 20-point course.

By taking advantage of relationships between existing material on Functional Programming and on Computation and Logic, and by trimming some material from Inf1-CL, we can present the existing material in a more unified way and free up space for more material on algorithms and on "computational thinking".

Taking advantage of the fact that some of the more advanced material in Inf1-FP is not currently examinable, we can provide a version of the course that replaces this material with introductory material on imperative programming, bringing students who have no previous programming experience up to speed for Inf1-OP in second semester and enabling that course to be taught to a more uniform group of students.

1b. Target audience and expected demand

[Describe the type of student the course would appeal to in terms of background, level of ability, and interests, and the expected class size for the course based on anticipated demand. A good justification would include some evidence, e.g. by referring to projects in an area, class sizes in similar courses, employer demand for the skills taught in the course, etc.]

This course would be compulsory for all Informatics students and would be accessible to students from outside Informatics, with no prerequisites. The expected class size would be the same as for the current Inf1-FP and Inf1-CL.

1c. Relation to existing curriculum

[This section should describe how the proposed course relates to existing courses, programmes, years of study, and specialisms. Every new course should make an important contribution to the delivery of our Degree Programmes, which are described at http://www.drps.ed.ac.uk/15-16/dpt/drps_inf.htm.

Please name the Programmes the course will contribute to, and justify its contribution in relation to courses already available within those programmes. For courses available to MSc students, describe which specialism(s) the course should be listed under (see http://web.inf.ed.ac.uk/infweb/student-services/ito/students/taught-msc-2015/programme-guide/specialist-areas), and what its significance for the specialism would be. Comment on the fit of the proposed course with the structure of academic years for which it should be offered. This is described in the Year Guides linked from http://web.inf.ed.ac.uk/infweb/student-services/ito/students.]

The proposed course would replace Inf1-FP and Inf1-CL. The material required by subsequent courses would be preserved. Introductory material on imperative programming would be provided in order to prepare students with no previous programming background for Inf1-OP. It is likely that material from the beginning of Inf1-OP can be adapted for use here, freeing up time in Inf1-OP for more advanced material and/or for more consolidation of the material that is taught.

1d. Resources

[While course approvals do not anticipate the School's decision that a course will actually be taught in any given year, it is important to describe what resources would be required if it were run. Please describe how much lecturing, tutoring, exam preparation and marking effort will be required in steady state, and any additional resources that will be required to set the course up for the first time. Please make sure that you provide estimates relative to class size if there are natural limits to its scalability (e.g. due to equipment or space requirements). Describe the profile of the course team, including lecturer, tutors, markers, and their required background. Where possible, identify a set of specific lecturers who have confirmed that they would either like to teach this course apart from the proposer, or who could teach the course in principle. It is useful to include ideas and suggestions for potential teaching duty re-allocation (e.g. through course sharing, discontinuation of an existing course, voluntary teaching over and above normal teaching duties) to be taken into account when resourcing decisions are made.]

The resources required would be about the same as the resources currently required by Inf1-FP and Inf1-CL: four hours of lectures per week, two hours of tutorials/exercise sessions per week, labs with demonstrators to provide assistance to students working on exercises. The examination would include both an on-line programming component and a pen-and-paper component.

Additional resources required are six lectures on introductory imperative programming and accompanying unassessed exercises.

SECTION 2 – COURSE DESCRIPTOR

[This is the official course descriptor that will be published by the University and serves as the authoritative source of information about the course for student via DRPS and PATH. Current course descriptions in the EUCLID Course Catalogue are available at www.euclid.ed.ac.uk under 'DPTs and Courses', searching for courses beginning 'INFR'*]*

2a. Course Title *[Name of the course.]*:

Informatics 1 – Introduction to Computation

2b. SCQF Credit Points:

[The Scottish Credit and Qualifications Framework specifies where each training component provided by educational institutions fits into the national education system. Credit points per course are normally 10 or 20, and a student normally enrols for 60 credits per semester. For those familiar with the ECTS system, one ECTS credit is equivalent to 2 SCQF credits. See also http://www.scqf.org.uk/The%20Framework/Credit%20Points.]

20

SCQF Credit Level:

[These levels correspond to different levels of skills and outcomes, see http://www.sqa.org.uk/files_ccc/SCQF-LevelDescriptors.pdf At University level, Year 1/2 courses are normally level 8, Year 3 can be level 9 or 10, Year 4 10 or 11, and Year 5/MSc have to be level 11. MSc programmes may permit a small number (up to 30 credits overall) of level 9 or 10 courses.]

8

Normal Year Taken: 1/2/3/4/5/MSc

[While a course may be available for more than one year, this should specify when it is normally taken by a student. "5" here indicates the fifth year of undergraduate Masters programmes such as MInf.]

1

Also available in years: 1/2/3/4/5/MSc

Different options are possible depending on the choice of SCQF Credit Level above: for level 9, you should specify if the course is for 3rd year undergraduates only, or also open to MSc students (default); for level 10, you should specify if the course is available to 3rd year and 4th year undergraduates (default), 4th year undergraduates only, and whether it should be open to MSc students; for level 11, a course can be available to 4th and 5th year undergraduates and MSc students (default), to 5th year undergraduates and MSc students, or to MSc students only]

2c. Subject Area and Specialism Classification:

[Any combination of Computer Science, Artificial Intelligence, Software Engineering and/or Cognitive Science as appropriate. For courses available to MSc students, please also specify the relevant MSc specialist area (to be found in the online MSc Year Guide at http://web.inf.ed.ac.uk/infweb/student-services/ito/students/taught-msc-2015/programme-guide/specialist-areas), distinguishing between whether the course should be considered as "core" or "optional" for the respective specialist area.]

Computer Science, Artificial Intelligence, Software Engineering, Cognitive Science

Appropriate/Important for the Following Degree Programmes:

[Please check against programmes from http://www.drps.ed.ac.uk/15-16/dpt/drps_inf.htm to determine any specific programmes for which the course would be relevant (in many cases, information about the Subject Area classification above will be sufficient and specific programmes do not have to be specified). Some courses may be specifically designed for non-Informatics students or with students with a specific profile as a potential audience, please describe this here if appropriate.]

All undergraduate Informatics degree programmes

Timetabling Information:

[Provide details on the semester the course should be offered in, specifying any timetabling constraints to be considered (e.g. overlap of popular combinations, other specialism courses, external courses etc).]

Semester 1

2d. Summary Course Description:

[Provide a brief official description of the course, around 100 words. This should be worded in a student-friendly way, it is the part of the descriptor a student is most likely to read.]

An introduction to concepts of programming, using a functional programming language, and to concepts of computation and specification using finite-state systems and propositional logic. These provide examples of the logical ideas of syntax and semantics and the computational ideas of structure and behaviour. Students learn to specify, model and solve small-scale problems succinctly and at an abstract level.

Course Description:

[Provide an academic description, an outline of the content covered by the course and a description of the learning experience students can expect to get. See guidance notes at: http://www.studentsystems.is.ed.ac.uk/staff/Support/User_Guides/CCAM/CCAM_Information_Captured.html

An introduction to concepts of programming, using the Haskell functional programming language, and to concepts of computation and specification, using finite-state machines and propositional logic. The use of sets, functions and relations to describe models of logic and computation. Programming using functions and data structures including lists and trees; case analysis, recursion and higher-order functions. Finite-state machines as a basic model of computation: deterministic and non-deterministic automata; regular expressions; acceptors; structured design of finite state machines. Propositional logic: truth tables; satisfiability; deduction. Applications from different areas will be used to illustrate and motivate the material.

Pre-Requisite Courses:

[Specify any courses that a student must have taken to be permitted to take this course. Pre-requisites listed in this section can only be waived by special permission from the School's Curriculum Approval Officer, so they should be treated as "must-have". By default, you may assume that any student who will register for the course has taken those courses compulsory for the degree for which the course is listed in previous years. Please include the FULL course name and course code].

Co-Requisite Courses:

[Specify any courses that should be taken in parallel with the existing course. Note that this leads to a timetabling constraint that should be mentioned elsewhere in the proposal. Please include the FULL course name and course code].

[Specify any courses that should not be taken in combination with the proposed course. Please include the FULL course name and course code].

Other Requirements:

[Please list any further background students should have, including, for example, mathematical skills, programming ability, experimentation/lab experience, etc. It is important to consider that unless there are formal prerequisites for participation in a course, other Schools can register their students onto our courses, so it is important to be clear in this section. If you want to only permit this by special permission, a statement like "Successful completion of Year X of an Informatics Single or Combined Honours Degree, or equivalent by permission of the School." can be included.]

SCE H-grade Mathematics or equivalent is desirable.

Available to Visiting Students: Yes/No

[Provide a justification if the answer is No.]

Yes

2e. Summary of Intended Learning Outcomes (MAXIMUM OF 5):

[List the learning outcomes of the course, emphasising what the impact of the course will be on an individual who successfully completes it, rather than the activity that will lead to this outcome. Further guidance is available from https://canvas.instructure.com/courses/801386/files/24062695]

On completion of this course, the student will be able to:

1. Use sets, functions and relations to create a simple mathematical model of a real-world situation and use the syntax and semantics of propositional logic to express simple constraints.
2. Solve simple programming tasks and define appropriate data types. Choose appropriate decompositions of given problems and compose corresponding functional programs from suitable function definitions, including their types.
3. Read and write programs that use basic list processing functions, list comprehensions, case analysis, recursion, and higher-order functions. Understand algorithms for searching and sorting. Document, test and debug programs.
4. Formalise simple propositional reasoning using various methods, including truth tables.
5. Design finite state acceptors for particular languages. Use regular expressions to search for simple patterns. Understand the relationship between finite state acceptors and regular expressions.

Assessment Information

[Provide a description of all types of assessment that will be used in the course (e.g. written exam, oral presentation, essay, programming practical, etc) and how each of them will assess the intended learning outcomes listed above. Where coursework involves group work, it is important to remember that every student has to be assessed individually for their contribution to any jointly produced piece of work. Please include any minimum requirements for assessment components e.g. student must pass all individual pieces of assessment as well as course overall].

A practical final exam will assess programming in Haskell. A mid-semester written class exam will assess progress towards learning to program in Haskell. A written final exam will assess understanding of propositional logic and finite state machines. Students' solutions to weekly unassessed formative exercises will be discussed in tutorial groups.

The material on advanced functional programming in Haskell and on introductory imperative programming will not be assessed. The latter material will be covered by unassessed formative exercises.

The marks from the practical and written final exams will be combined to give a single exam mark. Students are required to achieve a passing mark for the course as a whole; there is no requirement that they separately pass one or both of the exams.

Assessment Weightings:

Examinations: 95%

Coursework: 5%

Time spend on assignments:

[Weightings up to a 70/30 split between exam and coursework are considered standard, any higher coursework percentage requires a specific justification. The general expectation is that a 10-point course will have an 80/20 split and include the equivalent of one 20-hour coursework assignment (although this can be split into several smaller pieces of coursework. The Practical Examination category should be used for courses with programming exams. You should not expect that during term time a student will have more than 2-4 hours to spend on a single assignment for a course per week. Please note that it is possible, and in many cases desirable, to include formative assignments which are not formally assessed but submitted for feedback, often in combination with peer assessment.]

As in Inf1-FP and Inf1-CL.

Academic description:

[A more technical summary of the course aims and contents. May include terminology and technical content that might be more relevant to colleagues and administrators than to students.]

Functional programming is appropriate for a first university course on programming because it requires students to think about programming at a high level, allowing the emphasis to be put mainly on understanding fundamental concepts rather than mainly on coming to grips with programming language syntax and idiosyncrasies. Also, it puts students who have no programming experience on the same level as students with a great deal of programming experience, addressing the diversity of our intake. Simple set theory, finite-state machines and propositional logic are essential foundations for modelling and specifying problems and computations. Teaching these topics together with functional programming allows us to address all aspects of simple computational problem solving – modelling, specification, programming – in a single course, and to tie the topics in the course together by (for example) writing programs that manipulate propositional formulae.

[Provide a more detailed description of the contents of the course, e.g. a list of bullet points roughly corresponding to the topics covered in each individual lecture/tutorial/coursework. The description should not exceed 500 words but should be detailed enough to allow a

student to have a good idea of what material will be covered in the course. Please keep in mind that this needs to be flexible enough to allow for minor changes from year to year without requiring new course approval each time.]

- Set theory
- Functions and lists
- List comprehensions
- Recursion on lists
- Properties of functions
- Recursion on natural numbers; zip, infinite lists
- Select, take, drop, as applications of zip
- Higher-order functions: map, filter, fold
- Lambda expressions, sections, binding
- Algebraic data types
- Algebraic data types and expression trees
- Expression trees, Maybe type, union of types
- Introduction to big-O notation, efficiency of different representations of sets
- Binary search trees, balanced trees (AVL trees)
- Search in trees: breadth-first search, depth-first search, backtracking
- Sorting lists, divide and conquer
- Representation invariant, data abstraction
- Logical connectives, truth tables
- Tautology, satisfiable, contradiction, connection with circuits of switches, conversion to CNF
- Proof using deduction rules
- States, state transition function, deterministic vs non-deterministic
- Deterministic FSMs more formally, acceptors
- Language accepted by an FSM, regular languages, non-deterministic FSMs
- Regular expressions, converting FSMs to regular expressions
- Converting regular expressions to FSMs, regular expressions for search

During the final three weeks of the course, 6 lectures per week will be offered rather than the usual 4 lectures per week. Two of these will be compulsory. Two of the remaining 4 lectures will cover advanced topics in functional programming, for instance:

- Type classes
- IO and commands, do-notation, bind operator
- Monads, connection between list monad and list comprehension, applications
- Connections between logic and lambda calculus / functional programming

The other two of the remaining 4 lectures will cover introductory imperative programming, for instance:

- Objects, fields, methods
- Assignment, conditional, loops

Students will self-select between these streams according to previous programming experience and confidence; students who have no previous programming experience and who plan to take Inf1-OP will be advised to take the latter stream.

Relevant QAA Computing Curriculum Sections:

[Please see http://www.qaa.ac.uk/en/Publications/Documents/SBS-Computing-consultation-15.pdf to check which section the course fits into.]

[As Inf1-FP and Inf1-CL.]

Graduate Attributes, Personal and Professional skills:

[This field should be used to describe the contribution made to the development of a student's personal and professional attributes and skills as a result of studying this course – i.e. the generic and transferable skills beyond the subject of study itself. Reference in particular should be made to SCQF learning characteristics at the correct level http://www.sqa.org.uk/files_ccc/SCQF-LevelDescriptors.pdf].

[As Inf1-FP and Inf1-CL, plus teamwork, fostered through some use of pair programming.]

Breakdown of Learning and Teaching Activities:

[Total number of lecture hours and tutorial hours, with hours for coursework assignments.]

[The breakdown of learning and teaching activities should only include contact hours with the students; everything else should be accounted for in the Directed Learning and Independent Learning hours.

The total being 10 x course credits. Assume 10 weeks of lectures slots and 10 weeks of tutorials, though not all of these need to be filled with actual contact hours. As a guideline, if a 10-pt course has 20 lecture slots in principle, around 15 of these should be filled with examinable material; the rest should be used for guest lectures, revision sessions, introductions to assignments, etc. Additional categories of learning and teaching activities are available, a full list can be found at:

http://www.euclid.ed.ac.uk/Staff/Support/User_Guides/CCAM/Teaching_Learning.htm]

Lecture Hours: 40 hours

Seminar/Tutorial Hours: 20 hours

Supervise practical/Workshop/Studio hours: 20 hours

Summative assessment hours: 5 hours

Feedback/Feedforward hours: 4 hours

Directed Learning and Independent Learning hours: 111 hours

Total hours: 200 hours

You may also find the guidance on ‘Total Contact Teaching Hours’ and ‘Examination & Assessment Information’ at:
http://www.studentsystems.ed.ac.uk/Staff/Support/User_Guides/CCAM/CCAM_Information_Captured.html

Keywords:

[A list of searchable keywords.]

[As Inf1-FP and Inf1-CL.]

SECTION 3 - COURSE MATERIALS

3a. Sample exam question(s)

[Sample exam questions with model answers to the individual questions are required for new courses. A justification of the exam format should be provided where the suggested format non-standard. The online list of past exam papers gives an idea of what exam formats are most commonly used and which alternative formats have been http://www.inf.ed.ac.uk/teaching/exam_papers/.]

See past Inf1-FP and Inf1-CL exam papers.

3b. Sample coursework specification

[Provide a description of a possible assignment with an estimate of effort against each sub-task and a description of marking criteria.]

See past Inf1-FP class test papers.

3c. Sample tutorial/lab sheet questions

[Provide a list of tutorial questions and answers and/or samples of lab sheets.]

See past Inf1-FP and Inf1-CL tutorial exercises.

[Include anything else that is relevant, possibly in the form of links. If you do not want to specify a set of concrete readings for the official course descriptor, please list examples here.]

SECTION 4 - COURSE MANAGEMENT

4a. Course information and publicity

[Describe what information will be provided at the start of the academic year in which format, how and where the course will be advertised, what materials will be made available online and when they will be finalised. Please note that University and School policies require that all course information is available at the start of the academic year including all teaching materials and lecture slides.]

[As Inf1-FP and Inf1-CL.]

4b. Feedback

[Provide details on feedback arrangements for the course. This includes when and how course feedback is solicited from the class and responded to, what feedback will be provided on assessment (coursework and exams) within what timeframe, and what opportunities students will be given to respond to feedback.

The University is committed to a baseline of principles regarding feedback that we have to implement at every level, these are described at http://www.docs.sasg.ed.ac.uk/AcademicServices/Policies/Feedback_Standards_Guiding_Principles.pdf.

Further guidance is available from http://www.enhancingfeedback.ed.ac.uk/staff.html.]

[As Inf1-FP and Inf1-CL.]

4c. Management of teaching delivery

[Provide details on responsibilities of each course staff member, how the lecturer will recruit, train, and supervise other course staff, what forms of communication with the class will be used, how required equipment will be procured and maintained. Include information about what support will be required for this from other parties, e.g. colleagues or the Informatics Teaching Organisation.]

[As Inf1-FP and Inf1-CL.]

SECTION 5 - COMMENTS

[This section summarises comments received from relevant individuals prior to proposing the course. If you have not discussed this proposal with others please note this].

5a. Year Organiser Comments

[Year Organisers are responsible for maintaining the official Year Guides for every year of study, which, among other things, provide guidance on available course choices and specialist areas. The Year Organisers of all years for which the course will be offered should be consulted on the appropriateness and relevance on the course. Issues to consider here include balance of course offerings across semesters, subject areas, and credit levels, timetabling implications, fit into the administrative structures used in delivering that year.]

5b. BoS Academic Secretary

[Any proposal has to be checked by the Secretary of the Board of Studies prior to discussion at the actual Board meeting. This is a placeholder for their comments, mainly on the formal quality of the content provided above.]