TIG023, Logic, Algorithms and Data structures, 
7.5 higher education credits

First Cycle

This syllabus in English is the binding document.

1. Confirmation

The syllabus was confirmed by the Faculty Board of the IT University on 2007-02-14 and revised on 2007-12-04 to be valid from spring semester, 2009.

Field of education: Sciences.

Responsible department: Applied Information Technology.

2. Position in the educational system

The course TIG023, Logic, Algorithms, and Data structures, 7.5 higher education credits, is given as a compulsory course within the Software Engineering and Management Bachelor program.

3. Entrance qualifications

To be eligible for this course, the equivalent of 7,5 higher education credits in Object Oriented Programming with Java is required, in addition to high-school level qualifications in English and Algebra (Comparable to the courses English and Mathematics C defined by Skolverket). It is assumed that the student is familiar with the Programming language Java, some development environment for Java, development of graphical user interfaces in Java, with the organization of Java’s documentation, and with the use of Java’s built-in array data structure.

4. Course content

The course has three general themes: (1) discrete mathematics and its use for specifying, implementing, and documenting algorithms; (2) the role of algorithms in the design and development of programs; (3) the role of data structures in the implementation of algorithms. These themes are approached laterally such that the relationships between mathematical concepts, the realization of these concepts in a programming language, and the use of these concepts to solve concrete programming problems are highlighted. These general themes are supported by the study of sub themes from within the fields of mathematics and computer science, all selected to highlight the usefulness of
mathematics in programming: formal logic, complexity, sequences, functions and mappings, recursion, set theory, trees, data structures for sequences, data structures for trees, hash-tables, orderings, sorting algorithms, graphs, graph algorithms, data structures for graphs, matrices, finite state machines, regular expressions, optimization, data structures for 3D graphics.

These (and possibly also related) sub-themes are covered in 10-12 study items, each consisting of an introduction to sub-themes, selected chapters of the course literature study, and a programming problem that depends on topic knowledge within the studied sub-themes.

5. Learning outcomes

After completing the course, the student will have the following abilities:

1. do simple calculations within the following mathematical theories: statement logic, predicate logic, set theory, graph theory, matrix theory.
2. reads, specifies, and describes algorithms, at a higher level of abstractions than code,
3. reads and follows the mathematical descriptions of algorithms recurrent in engineering research literature,
4. identifies and chooses algorithms in the engineering research literature, given the nature of a computational problem,
5. implement such algorithms in a programming language with clean and efficient results,
6. present own algorithms using mathematical language and pseudo code,
7. improve the performance of inefficient programs, by choice of more appropriate data structures, and algorithms of better complexity.
8. solve application-level programming problems, with algorithmic content, in short time, by software reuse and teamwork.

6. Required reading

The course uses two complementary main texts:


7. Assessment

The course is examined by a compulsory written examination (for learning outcomes 1–3) and one voluntary programming problem for each item (for learning outcomes 4–8). The written exam is individual, whereas programming problems are carried out and reported on in groups, along with a clear indication of the relative contribution of the
group’s members to pass the course, demonstrable knowledge in 1-3 is sufficient; for a higher grade, the student must, in addition, demonstrate knowledge in 4-8 by substantially contributing to the programming problems handed in by the group. The results from the written examination and the programming problems are combined into a common grade for the course.

A student who has failed examination twice has the right to change examiner, unless weighty argument can be adduced. The application shall be sent to the board of the department and has to be in writing.

8. Grading scale
The grades are Pass with honour (VG), Pass (G) or Fail (U) is given after reviewing the written report. On request, grades are mapped onto an ECTS grading scale using a fixed, GU common mapping.

A student who has failed a test twice has to the right to change examiner. The application shall be sent to the head of the department and has to be in writing.

9. Course evaluation
The course will be evaluated by the course responsible using a survey. The results of the evaluation will be communicated to the students and will function as a guide for the development of the course.

10. Additional information
Lectures, programming problems, and the written exam will be given in English. Teams may use either Swedish or English as working language. Handins are written in English, whereas answers at the written exam may be written in either English or Swedish.

The course’s overall pedagogical strategy is problem based learning.