Budapest University of Technology and Economics,
Faculty of Electrical Engineering and Informatics **Technical Informatics**

Course Datasheet
and requirements
17th December, 2003

Declarative programming

2.	Code	Semester	Requirements	Credits	Language	Semester
	???	4	4+0+0v	5	English	2/2

3. The person responsible for the course

Name
dr. Péter Szeredi

4. Lecturers

Name
dr. Péter Hanák
dr. Péter Szeredi

5. Prerequisites/previous knowledge required:

General programming skills, Technology of programming, Mathematical logic

6. Compulsory order of studies:

---

7. Aim of the course:

The aim of the course is to get the students acquainted with the idea of declarative programming. In contrast with imperative programming, covered by earlier and other courses, declarative programming means a different point of view, which is explained during the course.

8. Detailed syllabus of the course

Comparison of the imperative and declarative (both logic and functional) programming paradigms.

Functional programming, using the SML language:

Expressions, values, types. Simple and compound types (tuples, records, lists, trees, etc.). Binding. Type inference system. Pattern matching. Functions, higher-order functions, partial functions. Type parameters, polymorphism. Recursion, iteration. Lazy lists. Abstract types. Structures, signatures, functors. SML Basic Library. Programming techniques. New directions in functional programming.

Logic programming, using the Prolog language:

The basic elements of the language: predicates, control structures, compound data structures, disjunctions, operators, lists. Built-in predicates. Programming techniques. Advanced features and their application, modularity, exception handling. New directions in logic programming.

9. Methods of education

Lectures, consultations, computer exercises.

10. Requirements

a) Prerequisites

Basic level knowledge in logic, theoretical and practical programming skills.

b) Requirements during the semester

Obligatory: to pass a mid-term test
Recommended: students are strongly recommended to use the Web-based Electronic Teaching aSsistant (ETS) as well as to solve the minor and major programming assignment(s) set during the semester, for both SML and Prolog languages.

c) The date, evaluation and acceptance conditions of the mid-term test

The mid-term test will be held on the week specified by the course schedule. Students are required to master only those parts of the curriculum that were presented or prescribed until the end of the week preceding the mid-term test.

Students can re-take the mid-term test if they have failed it, or if they want to improve their results. There is a re-take test near the end of the semester and an additional one – which takes the form of a repeat exam – during the first three weeks of the examination period. The requirements for the re-take mid-term tests are the same as for the normal mid-term test.

During the mid-term test students are not allowed to use anything, but pen and paper (e.g. use of lecture notes, calculators, cellular phones, etc. is disallowed). The mid term test is accepted, if the student achieves 40-40% of the maximum points for **both** the Prolog and SML parts. The only exception comes into force if a student has a signature from a previous semester. In this case the 40% limit does not apply.

The result of the mid-term test (which is defined as the highest score achieved on the normal and the re-take mid-term tests) accounts with a weight of **15%** in calculating the final examination mark.

d) Evaluation and acceptance conditions of the programming assignments

The major assignment will be announced in the week indicated in the course schedule. The programs for solving the assignment task have to be written in SML and/or Prolog, and have to be handed in electronically via the Electronic Teaching aSsistant (ETS), together with the appropriate documentation. Further details on this can be found in the assignment description on the subject homepage.

We use at least three distinct sets of test cases for testing the students' programs. The first set is given at the time of announcement in order to be used by the students in the process of program development. The second set is used when the students hand in their solutions. The results of this test are sent back to the students via email. The students can hand in as many versions of their programs as they wish (as long as they are within the deadline), but only the latest version will be used for the final test. The final test is run on the third set of test cases, which has a similar difficulty level as the earlier ones.

A set of test cases usually contains ten or more test cases. The student's program gets credited for a given test case if it provides the appropriate answer within the prescribed period of time. The programs should be well written and easily readable for others. We give points for this as well as for the documentation, which is also to be submitted electronically in ASCII, HTML, PS, or PDF format. Further details about the documentation requirements can be read in the announcement of the assignment.

The programs have to solve at least 40% of the test cases in order for the assignment to be accepted. Students get points for both the SML and Prolog programs. The results of the major assignment are taken into account in the final examination mark with a weight of **15%** (7.5% for each language).

The best solutions for the major assignment (which solve all the test cases and have well written documentation) automatically participate in the so-called "ladder contest". These programs will be tested against bigger and more difficult test cases. The writers of the programs, which achieve highest places in the ladder contest, get extra points.

Throughout the semester several minor assignments will be announced. For correct solutions students can get extra points. Further information about how to hand in minor assignments can be found on the homepage.

The points for the ladder contest and minor assignments will be considered when calculating the final mark.

The minor and major programming assignments are non-compulsory. They can not be accepted after the given deadline.

e) Use of the Electronic Teaching aSsistant

The Web-based ETS may significantly help the students to gain experience in declarative programming. Tips on how to use the system can be found on the homepage of the course.

f) Condition of acquiring the signature

The acceptance condition is to reach 40-40% in **both** the Prolog and SML parts of the mid-term test. This can also be achieved in the re-take mid-term tests.

g) Eligibility for the exam

Students can register for the exam if they have a valid signature. It does not matter when they actually have acquired the signature as long as this has happened no more than 4 semesters before the current semester.

h) The exam

The exam is oral, combined with written exercises. The condition for passing the exam is to achieve at least **40%** of the maximal points from **both** languages. Further details about the exam will be provided during the semester.

i) The final mark

The final mark is calculated from the points obtained during the exam (with a weight of 70%), the best of the mid-term tests (with a weight of 15%) and the (non-compulsory) major programming assignment (with a weight of 15%). The points given for the minor assignments and/or for the ladder competition further improve the final score. A student, who solves no programming assignment, can only have 85% of the maximum score.

Important note: the points for the mid-term test(s) and the assignments are taken into account only if they were obtained during the semester in question. So, a student who has a signature from a previous semester and writes no mid-term test and/or no assignments, can only have 70% of the maximum score.

j) Use of disallowed devices or assistance

On the mid-term test students are not allowed to use anything but pen and paper. Students who violate these rules face a penalty according to the valid Faculty and University regulations.

Students have to solve the minor and major programming assignments independently

from each other. Except for the informal exchange of ideas, no assistance from other persons can be used, no program code or code fragment can be shared. Automatic tools are used to check the similarity of the programs.

If a student passes or receives programs or code fragments, his or her assignments are considered invalid.

In case of issues unregulated here the Academic and Examination Regulations take force.

11. Re-take possibilities

The re-take mid-term test will be held in the final weeks of the semester. An additional re-take possibility – which takes the form of a repeat exam – is possible during the first three weeks of the examination period. The requirements for the re-take mid-term tests are the same as in the case of the normal mid-term test.

The non-compulsory minor and major programming assignments can not be accepted after the given deadline.

12. Consultation possibilities

Students who have subscribed to the mailing list of the course have the possibility to ask questions during the whole semester. These questions will be answered continually. It is also possible to have a personal consultation with the lecturers, which has to be requested in advance.

13. Textbooks and literature

The textbooks for the course can be bought in a way described on the course homepage. The textbooks are also available in electronic formats.

a.Szeredi Péter, Benkő Tamás: **Deklaratív programozás.** Oktatási segédlet. *Bevezetés a logikai programozásba.* Számítástudományi és Információelméleti Tanszék - IQSOFT Rt., Budapest, 2001 (in Hungarian)
b.Hanák D. Péter: **Deklaratív programozás.** Oktatási segédlet. *Bevezetés a funkcionális programozásba.* Irányítástechnika és Informatika Tanszék, Budapest, 2001 (in Hungarian)

Further recommended literature:

a.Jeffrey D. Ullman: *Elements of ML Programming.* Prentice Hall, 1993, ISBN: 0-13-184854-2
b.Richard Bosworth: *A Practical Course in Functional Programming using Standard ML.* McGraw-Hill, 1995. ISBN 0-07-707625-7
c.Colin Myers, Chris Clack, Ellen Poon: *Programming with Standard ML.* Prentice Hall, 1993, ISBN 0-13-722075-8
d.Robert Harper: *Programming in Standard ML.* School of Computer Science, Carnegie-Mellon University, 1986-2000. <http://www.cs.cmu.edu/~rwh/introsml/>
e.Lawrence C. Paulsson: *ML for the Working Programmer.* Cambridge University Press, 1991, ISBN 0-521-39022-2
f.Farkas Zsuzsa, Futó Iván, Langer Tamás, and Szeredi Péter: *M-Prolog programozási nyelv.* Műszaki Könyvkiadó, Budapest, 1989.
g.Márkusz Zsuzsa: *Prologban programozni könnyű.* Novotrade, Budapest, 1988.
h.Richard A. O'Keefe: *The Craft of PROLOG.* MIT Press, 1990. ISBN 0-262-15039-5
i.Leon Sterling, Ehud Shapiro. *The Art of Prolog.* Advanced Programming Techniques, 2nd Edition, MIT Press, 1994, ISBN 0-262-19338-8
j.David H. D. Warren: *Logic Programming and Compiler Writing.* In Software-Practice and Experience 10(2): 97-125, 1980.

14. The average amount of work necessary to complete the course

(The table contains the timesheet with a suggested partitioning of the whole course time. The expected work from a student for one credit point is 30 hours per semester.)

Lectures	60
Preparation during semester	15
Preparation for the mid-term test	5
Programming assignments	30
Other independent preparation	10
Preparation for the exam	30
Altogether	150

15. Other useful information about the course

Homepage: <http://dp.iit.bme.hu>.
Mailing list: <http://www.iit.bme.hu/mailman/listinfo/dp-l/>.

Information about the Prolog and SML interpreters (which can be used for practice and to solve the programming assignments) and how to download and install them are given on the homepage in detail. The homepage also contains information about the programming assignments, the mid-term tests, the Electronic Teaching aSsistant, etc.

16. The syllabus of the course was developed by

Department	name	position
Department of Control Engineering and Information Technology	dr. Péter Hanák	senior lecturer
Department of Computer Science and Information Theory	dr. Péter Szeredi	associate professor