STUDY MODULE DESCRIPTION FORM

Name of the module/subject	Code
Introduction to Computing	**1010511311010510189**

Field of study	Profile of study (general academic, practical)	Year /Semester
Computing	**general academic**	**1 / 1**
Elective path/specialty	Subject offered in:	Course (compulsory, elective)
-	**English**	**elective**

Cycle of study:	Form of study (full-time,part-time)
First-cycle studies	**full-time**

No. of hours	No. of credits
Lecture: **30** Classes: **-** Laboratory: **30** Project/seminars: **-**	**5**

Status of the course in the study program (Basic, major, other)	(university-wide, from another field)
major	**from field**

Education areas and fields of science and art	ECTS distribution (number and %)
technical sciences	**5 100%**
Technical sciences	**5 100%**

Responsible for subject / lecturer:

dr hab. inż. Jerzy Nawrocki, prof. nadzw.
email: jerzy.nawrocki@put.poznan.pl
tel. 665-2980
Faculty of Computing
ul. Piotrowo 3 60-965 Poznań

Prerequisites in terms of knowledge, skills and social competencies:

1	**Knowledge**	The knowledge on the level IV of the Polish Qualification Framework (confirmed by passing maturity exam) is expected.
2	**Skills**	Ability of logical thinking and attention focus are required.
3	**Social competencies**	Honesty, responsibility, interest in acquiring knowledge, creativity, propriety, respect.

Assumptions and objectives of the course:

To present to the students an overview of the field of Computing, including the relationship between different areas of Computing; particular attention is paid to practical aspects of Computing and the directions for development.

Study outcomes and reference to the educational results for a field of study

Knowledge:

http://www.put.poznan.pl/

1. The student has a general knowledge about computer hardware including its alternative forms (Turing Machine, DNA computing) and the paradigm of imperative programming (C i assembly-level programming) along with its ?competitors? (rule-based programming, regular expressions) - [K1s_W4]

2. The student has general knowledge about basic data structures (dynamic storage allocation, graphs, trees and lists, stacks, priority queue); notations used to describe computational complexity of algorithms (big O) and intractability (the halting problem); elements of numerical methods (numerical stability, Horner?s method, Maclaurin series and their application, Newton?s method) - [K1s_W4]

3. The student has general knowledge in the area of concurrent computations and their synchronization (POSIX, semaphores, the producer-consumer problem, the readers and writers problem); computer networks and cybersecurity; the relational model of data and SQL; microcontrollers and their applications; Soft. Eng. (software dev. life cycles, some UML diagrams and their application) - [[K1s_W4]

4. The student is aware of some alternative computing environments (e.g. DNA computing), post-imperative paradigms of programming (e.g. Board Programming), computer networks (the possible impact of the 5G technology) and cybersecurity (homomorphic encryption) and connections of Computer Science with Control Engineering, Robotics, Electronics, Mathematics, Communications, and Management - [K1st_W5]

5. The student has basic knowledge about software development lifecycles (the waterfall model, its weaknesses and possible countermeasures including agile methods) - [K1st_W6]

6. The student has knowledge about the principles of effectiveness proposed by Stephen Covey, ACM Code of Ethics, digital crime (e.g. ransomwere) and specificity of mission-critical systems (e.g. the notion of fail-safe) - [K1st_W8]

7. The student has basic knowledge of patents, copyright and related rights as well as the law on personal data protection (GDPR) - [Kst1_W11]

Skills:

1. The student is able to properly use ICT at various stages of IT projects implementation - [K1st_U2]

2. The student is able to discuss social, law, and economic aspects of IT - [K1st_U5]

3. The student can describe, using the big O notation, computational complexity of simple algorithms (e.g. heap sort) - [K1st_U8]

4. The student can work in a small team (up to 4 people) - [K1st_U18]

Social competencies:

1. The student understands that in the field of IT the knowledge and skills quickly become obsolete - [K1st_K1]

2. The student is aware of potential social and economic consequences resulting from malfunctioning IT systems - [K1st_K2]

Assessment methods of study outcomes

The learning outcomes concerning Knowledge Units W4, W5, W6, W8, W11, and Skills U2, U5, U18 will be checked in 3 tests (individual work), 2 contests (teams composed up to 4 students) and final exam (individual work). The learning outcomes U5, K1 and K2 will be evaluated during laboratory classes. The skills concerning U18 will be verified during the mentioned team contests.

Interim scores:

Each test will contain a few pairs of similar tasks. To pass a test consisting of N pairs of tasks, at least N-1 of them must be passed, and to pass a pair at least of the tasks of the pair must be passed. The students can bring with them their notes on an A4 sheet of paper. Smartphones, calculators and any other devices are not allowed.

Each team contest will contain several (up to 12) tasks. A task will be assigned a score within the range 0-10, and the result of each contest is relative to the maximum possible score, i.e. it can bring up to 100%. Each contest is an open-book exam (any notes or books are allowed except electronic devices).

Let Z1, Z2 denote results of the team concerning the first and second contest. The sum T1+T2 is used to create a ranking of the teams. Team?s grade G depends on the position of the team in the ranking. First 10% of the teams get the highest grade 5.0 (A), nest 20% get 4.5 (B), next 30% - grade 4.0 (C), and next 20% gets 3.5 (D). The last 20% of the teams have the N/A grade (Not Available). The grade of the team is ?inherited? by each team member. If a person does not participates in the contest, her/his grade for this contest is N/A.

The exam (and the second chance exam) will consists of test-like tasks and more difficult tasks resembling the contests. Let Tz and Tp denote the (binary) result of test part of the primary and second chance exam, respectively. Moreover, let Gz and Gp the grade obtain from the advanced part of the primary and second-chance exam, respectively (i.e. 5.0, 4.5, 4.0, 3.5, 30, or 2.0).

Final grade:

Let T1, T2, T3 denote logical variables which have the true value if the corresponding test has been passed. The final grade is defined by the following rules:

T1 and T2 and T3 => max{ 3.0, G, Gz }

not (T1 and T2 and T3) => (Tz => max{3.0, Gz})

not (T1 and T2 and T3) and not Tz (Tp => max{3.0, Gp})

not (T1 and T2 and T3) and not Tz and not Tp => 2.0 (F)

Course description

http://www.put.poznan.pl/

Stephen Covey?s principles of highly effective people; basic instructions of imperative languages (input-output, conditionals, loops). Digital systems (gates, decoders, multiplexers, adders, R-S flip-flops, registers and their addressing). Von Neumann concept. Two?s complement. Basic assembly-level instructions (arithmetic, conditional and unconditional jumps). Turing Machine. DNA computing. Data structures available at the hardware level (memory, stack) and at the level of C-like languages (arrays, records). Parameter passing. Pointers and dynamic storage allocation. Lists, trees, and graphs. Heap and its usage for sorting. Big O notation and computational complexity of algorithms. The limits of computability: fast growing functions and the halting problem. Machine-level representation or real numbers and the problem numerical stability. Horner?s method. Maclaurin?s series and their applications. Newton?s method and square root. The POSIX Standard, Concurrent computations and semaphores. The producer-consumer problem, the readers and writers problem. The TCP/IP protocol socket-level programming. Cybersecurity. Introduction to cryptography (DSA and RSA encryption, digital signature, homomorphic encryption). Microcontrollers 8051: architecture and programming basics (Arduino C). Mission-critical systems. Relational data model and introduction to SQL. Regular expressions. Introduction to Software Engineering (project lifecycle and its phases, functional requirements and use cases, architecture, testing. Selected UML diagrams. Ethical aspects (ACM code of conduct) and legal aspects (copyright law, patents, personal data protection).

Laboratory classes follow the lectures. Students solve tasks concerning the topics presented at the lecture and also do some programming.

Basic bibliography:

1. The C Programming Language, B.W. Kernighan, D.M. Ritchie, Prentice Hall, 1978.
2. The Essence of Digital Design, B. Wilkinson, Prentice-Hall, 1997.
3. Beej's Guide to Network Programming, 2016, http://beej.us/guide/bgnet/pdf/bgnet_A4.pdf

Additional bibliography:

1. 7 Habits of Highly Effective People, S. Covey, Rebis, 2003

Result of average student's workload

Activity	Time (working hours)
1. Participation in lectures	30
2. Participation in laboratory classes	30
3. Preparation to laboratory classes	20
4. Self-study	50
5. Preparation for the exam and participation in it.	10

Student’s workload

Source of workload	hours	ECTS
Total workload	140	5
Contact hours	62	2
Practical activities	40	2

http://www.put.poznan.pl/