Course Outlines
1st Semester
WEB DEVELOPMENT 1
Objectives
The course provides an introduction to front-end web development. The course covers the foundational building blocks of the Web and user interface design fundamentals for building websites and web applications. After the course, students will be able to understand and develop the front-end of any kind of websites and web applications.

Course learning outcomes
1.Understand the foundations of front-end development.

2.Identify the key components of web technologies.3.Judge and support best practices in web development.
Description
1. Getting started 1.1. Front-end roles: Design, develop, create, animate1.2. Languages: HTML, CSS, JS1.3. Setup: OS, browsers, editors2. How the web works 2.1. History: Internet, web browsers2.2. URIs: Scheme, conventions3. Key concepts to know 3.1. Methodologies: Progressive enhancement, graceful degradation, RWD3.2. Accessibility: Types of impairments4. HTML Overview 4.1. Defaults: Formatting, doctypes, stylesheets4.2. Semantic markup: Microdata, microformats4.3. Entities: Tags, attributes5. Marking up text 5.1. Structural elements: Metadata, sections, text, lists, tables, form, media6. Adding links 6.1. Working with URLs: Link types, attributes, fragments7. Adding images 7.1. Concepts: Formats, raster/vector, resolution, interlacing, transparency7.2. Optimization: Compression, dithering7.3. SVG: Animation, filters8. Tables 8.1. Markup notation: Semantics, controlling structure9. Forms 9.1. Markup notation: Elements, attributes9.2. Sending data: GET, POST, encodings10. HTML510.1. Overview: Features, browser wars10.2. APIs: Media Player, History, Storage, Drag and Drop, Canvas11. Cascading Style Sheets 11.1. Authoring: Formats, structure11.2. Inheritance: Tree matching, specificity11.3. Selectors: IDs, classes, conflicting styles, contextual and pseudo-selectors12. Formatting text 12.1. Fonts: Properties, family, style, color12.2. Measurement units: Absolute (px, pt) and relative (em, ex)13. Colors and backgrounds 13.1. Color models: RGB, RGBA, HSV, opacity13.2. Images: Properties, notation13.3. Gradients: Linear and radial gradients14. Thinking inside the box 14.1. Box model: Sizing, overflow, margin, padding, borders15. Floating and positioning 15.1. Layout flow: Containers, clears, stacking order16. Page layout 16.1. Types: Fixed, fluid, elastic, hybrid16.2. Methodologies: Common patterns, Flexbox, Grid system17. Transitions, transforms, and animation 17.1. Fundamentals: Functions, keyframes, properties18. CSS techniques 18.1. Tips: Resets, sprites18.2. Preprocessors: Variables, functions18.3. Responsive web design: Media queries
Assessment
Assessment modality: Coding assignments (continuous evaluation) and multiple-choice quiz exam.

Assessment tasks

Task 1: Coding assignments (40%)

Grading scheme: 20 points (0-20)

Objectives: Students will learn how to implement business requirements through hands-on exercises using JavaScript, HTML, and CSS.

Assessment rules: Every week, students must submit a simple programming exercise based on the contents taught in the course. Students must follow a series of specifications about each exercise, provided in Moodle. Students have up to 1 week to submit. Exercises must be submitted only via Moodle. No late submissions are allowed, unless “force majeure” causes; e.g. sickness, accident, etc. for which an authoritative certificate must be provided (e.g. a certificate signed by a medical doctor or a police representative).

Assessment criteria: There are 10 coding assignments in total, and each will be graded using a 0-10 point scale via an automated testing platform. Students can access such a platform at any time before each submission deadline. Students must get an average grade of at least 4.5 points (out of 10) in this part in order to qualify for the final exam. If a student scores less that 4.5 points in this part, the final grade in the course will be the minimum grade achieved in either the coding assignments or the final exam. Any form of cheating or plagiarism will be penalized with a grade of 0 in this part, which will make the student to fail the whole course. The grade in this part will be retained until the student passes the final exam, unless the student decides to repeat the course in the next academic year. There is no retaking option for this part within the same academic year, which means that a student who fails this part must enroll again in the course in the next academic year. Retaking students must inform the course responsible whether they want to keep their grade in this part from the previous year.Task 2: Final exam (60%)

Grading scheme: 20 points (0-20)

Objectives: Students will demonstrate that they have acquired the required competencies according to the theory covered in the course.

Assessment rules: Only students who got at least 4.5 points in the coding assignments should take the final exam, otherwise they are advised to de-enrol from the final exam in order to avoid missing an official attempt. Students can bring an A4 paper “cheat sheet” written on both sides. No other supporting material is allowed, i.e. no books, no mobile devices, no laptops, etc.

Assessment criteria: The exam is a paper-based quiz with 30 questions that must be solved in 45 minutes. Each question comes with 4 possible answers, among which only one is the right one. A student must get at least 4.5 points (out of 10) in the final exam in order to pass the course. If a student scores less that 4.5 points in this part, the final grade in the course will be the minimum grade achieved in either the coding assignments or the final exam. Any form of cheating or plagiarism will be penalized with a grade of 0 in this part, which will make the

student to fail the whole course. The grade in this part will not be retained for the next academic year. Failing students can retake the exam in the next examination period or enroll again in the course the next academic year.

Note

Syllabus☒Yes☐NoRemarks:Available on MoodleLiterature list☒Yes☐NoRemarks:Reference texbook:•J. Robbins. 2012. Learning Web Design. O'Reilly Media, 4th ed.Recommended books:•R. Anquetil. 2019. Fundamental Concepts for Web Development, 1st ed.•M. Haverbeke. 2018. Eloquent JavaScript, 3rd ed.•S. Krug. 2000. Don't Make Me Think, 3rd ed.Moodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=5689 Other, please specify:Class attendance is not required but highly encouraged.

ANALYSIS 1
Objectives
This course is an introductory analysis course, covering basics on numbers, sequences, and functions.

The course aims to introduce students to these notions, including formal proofs, techniques and to give them necessary tools for the understanding of mathematics and the mathematical language of science.

Course learning outcomes
At the end of the course the student will:-remember further mathematical knowledge. -apply analytic and critical thinking.-understand and practice deep mathematical concepts that will be used throughout their scientific career.-apply techniques from calculus to compute.
Description
This course is an introductory analysis course, covering basics on numbers, sequences, and functions. There is both a calculus component (computations of limits of sequences, derivatives, function approximations, limits of sequences) and more theoretical analysis.This latter portion is sizeable part of the course where the students are introduced to proof techniques and the importance of rigorous arguments which are foundational and transversal in mathematics and science.
Assessment
Assessment modality: Combined assessmentAssessment tasksTask 1: Written exam (60%)Grading scheme: 20 points (0-20)Objectives: Test concepts learned during the yearAssessment rules: Students cannot use calculators, electronic devices, or notes.Assessment criteria: Graded out of 20 with points for each exerciseTask 2: Written exam (20%)Grading scheme: 20 points (0-20)Objectives: Test concepts learned during the yearAssessment rules: These are partial exams during the semester (there are several of them). Students cannot use calculators, electronic devices, or notes.Assessment criteria: Graded out of 20 with points for each exerciseTask 3: Take-home assignment (10%)Grading scheme: 20 points (0-20)Objectives: Practice conceptsAssessment rules: Students work from home and can use course materials.Assessment criteria: Graded out of 20 with points for each exerciseTask 4: Written exam (10%)Grading scheme: 20 points (0-20)Objectives: QuizzesAssessment rules: There are 2 mandatory quizzes. Students cannot use calculators, electronic devices, or notes.Assessment criteria: Graded out of 20 with points for each exerciseTask 5: Written exam – RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: The final grade, in case of a retake exam, is based 100% on the exam grade. (Performance on quizzes, homework, midterms, and exercises is not taken into account.)Assessment rules: Students cannot use calculators, electronic devices, or notes.Assessment criteria: Graded out of 20 with points for each exercise
Note
Course materials

Syllabus☒Yes☐NoRemarks:Literature list☒Yes☐NoRemarks:Course notesMoodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=322

DISCRETE MATHEMATICS 1

Objectives
Discrete structures are foundational material for computer science. Relatively few computer scientists will be working primarily on discrete structures, but many other areas of computer science require the ability to work with concepts from discrete structures. The discrete structures covered in this introduction include important material from such areas as set theory, logic, graph theory, and number theory.

Course learning outcomes
By the end of the course students will be able to:-Use logical notation to define and reason about fundamental mathematical concepts, such as sets, relations, functions, and integers.-Evaluate elementary mathematical arguments and identify fallacious reasoning (not just fallacious conclusions).-Synthesise induction hypothesis and simple induction proofs.-Prove elementary properties of modular arithmetic and explain their applications in Computer Science, for example, in cryptography.-Apply graph theory models of data structures and state machines to solve problems of connectivity and constraint satisfaction, for example, scheduling.-Apply the method of invariants and well-founded ordering to prove correctness and termination of processes and state machines.-Efficiently work on problems in a small team with fellow students. -Manage their time and tasks in a responsible way.
Description
* Week 1 – Week 4: Proof techniques* Week 5 – Week 6: Introduction to Logic* Week 7: Mid-Term Exam* Week 8 – Week 9: Sets, Relations, and State Machines* Week 10 – Week 12: Graph Theory* Week 13 – Week 14: Number Theory and modular Arithmetic* Final Exam* Retake Exam
Assessment
Assessment modality: Combined assessment

Assessment tasks

Task 1: Written exam – Mid-Term (50%)Grading scheme: 20 points (0-20)Objectives: To test the students their knowledge of the first half of the material i.e. week 1-6 Proof techniques and Logic.Assessment rules: Students are allowed a cheat sheet using 1 side of 1 a4 page. Beside this, students can only bring pen and pencil. Final answers should be written using pen (blue/black).Assessment criteria: The answers to the exercises are graded based on correctness completeness and clarity. (see 9 points for writing good proofs in proof guide on moodle)Task 2: Written exam – Final Exam (50%)Grading scheme: 20 points (0-20) Objectives: To test the students their knowledge of the second half of the material i.e. week 7-13: sets, relations and state machines; graph theory; number theory and arithmetic.Assessment rules: Students are allowed a cheat sheet using 1 side of 1 a4 page. Beside this, students can only bring pen and pencil. Final answers should be written using pen (blue/black).Assessment criteria: The answers to the exercises are graded based on correctness completeness and clarity. (see 9 points for writing good proofs in proof guide on moodle)

In order to receive a final grade the midterm exam and final exam need to be completed.

Task 3: Written exam – RETAKE EXAM (100%)Grading scheme: 20 points (0-20) Objectives: Students who did not get an average of at least 10 points on the mid-term combined with the final exam are entitled to sit to the retaken test of the course. This test allows the student another opportunity to pass the course by obtaining at least 10 points on this retake exam. The retake exam covers all course material.Assessment rules: Students are allowed a cheat sheet using 1 side of 1 a4 page. Beside this, students can only bring pen and pencil. Final answers should be written using pen (blue/black).Assessment criteria: The answers to the exercises are graded based on correctness completeness and clarity. (see 9 points for writing good proofs in proof guide on moodle)
Note
Course materials

Syllabus☒Yes☐NoRemarks:Available on the Moodle page.Literature list☒Yes☐No

Remarks:-Lecture notes available on the Moodle page.https://moodle.uni.lu/pluginfile.php/2227981/mod_resource/content/1/Lecture%20Notes%20-%20MIT%20Discrete%20Math.pdfWe use the version from 2018.

-A Guide for making proofs. Available on the moodle page.Moodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=323 Other, please specify:We based this course on the MIT course Mathematics for Computer Science: https://ocw.mit.edu/courses/6-042j-mathematics-for-computer-science-fall-2010/

LINEAR ALGEBRA 1

Objectives
Fundamental notions of Linear Algebra.

Course learning outcomes
The students recall and are able:•To formulate the basic notions of linear algebra. •To solve small linear algebra problems by performing calculations and applying algorithms.•To understand small proofs and to look for examples and counterexamples.
Description
Systems of linear equations and matrices; determinants; Euclidean vector spaces (especially in dimension 2 and 3); general vector spaces (linear independence, span, basis).
Assessment
Assessment modality: Combined assessment

Assessment tasks

Task 1: Written exam – Midterm exam 1 (15%)Grading scheme: 20 points (0-20)Objectives: Written test with exercises.Assessment rules: Only 6 pages of personal notes are allowed. No calculators.Assessment criteria: The tests are graded. The average of the two best midterms counts as mark.Task 2: Written exam – Midterm exam 2 (15%)Grading scheme: 20 points (0-20)Objectives: Written test with exercises.Assessment rules: Only 6 pages of personal notes are allowed. No calculators.Assessment criteria: The tests are graded. The average of the two best midterms counts as mark.Task 3: Written exam – Midterm exam 3 (15%)Grading scheme: 20 points (0-20)Objectives: Written test with exercises.Assessment rules: Only 6 pages of personal notes are allowed. No calculators.Assessment criteria: The tests are graded. The average of the two best midterms counts as mark.Task 4: Written exam – Final exam (70%)Grading scheme: 20 points (0-20)Objectives: Written test with exercises.Assessment rules: Only 6 pages of personal notes are allowed. No calculators.Assessment criteria: The test is graded.Task 5: Oral exam – optional Grading scheme: 20 points (0-20)Objectives: Oral exam (in case of justified absence to >1 midterms or as additional evaluation in case of fraud suspicion)Assessment rules: Discussion with questions (theoretical questions and exercises)Assessment criteria: Correct and partially correct answers contribute positively to the grade.Task 6: Either final written exam (100%) or the same modalities as the regular students (to be selected by each retaking student at the beginning of each semester)Grading scheme: 20 points (0-20)Objectives: Written test with exercises.Assessment rules: Only 6 pages of personal notes are allowed. No calculators.Assessment criteria: The test is graded.
Note
Syllabus☒Yes☐NoRemarks:Moodle pageLiterature list☒Yes☐NoRemarks:Moodle pageMoodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=324 Other, please specify:

PROGRAMMING FUNDAMENTALS 1

Objectives

This course introduces the fundamentals of programming using the Python programming language. This is not primarily a Python programming course but rather a discussion of the fundamental concepts underlying computation using Python code examples as illustration. At the same time enough of the Python language is covered for the students to be able to tackle non-trivial problems (e.g., in the context of projects). This introductory course forms the basis for more advanced courses on algorithms and object-oriented programming.

Course learning outcomes

Upon completion of this course the student should be able to:•move from a problem description to a Python program by successively reducing the level of abstraction with the help of pseudo-code.•document the implementation choices. •make use of available data types and program libraries.•extend and adapt code written by other programmers.•test and debug computer programs.

Description

1. Introduction to computational problem solving and the Python programming language.2. Basic syntax and semantics of Python.3. Functions and modules.4. Problem solving and recursion.5. Structured types and function objects.6. Files and exceptions.7. Testing.8. Debugging.9. Iterators and generators.10. Floating-point numbers.11. Introduction to object-oriented programming in Python.12. Introduction to popular libraries: mathplotlib, NumPy and Pandas.13. Introduction to software engineering.

Assessment

Assessment modality: Combined assessmentAssessment tasksTask 1: Written exam (50%)Grading scheme: 20 points (0-20)Objectives: The objective of this final exam is to test the student’s understanding of the course material.Assessment rules: The student has to answer questions with pencil and paper. This is a closed-book exam. No cheat sheet allowed.Assessment criteria: The student must answer the stipulated questions in a way that clearly demonstrates understanding of underlying concepts.Task 2: Written exam (50%)Grading scheme: 20 points (0-20)Objectives: To test the student’s ability to implement Python programs for concrete problems.Assessment rules: The student has to solve programming tasks with pencil and paper. This is a closed-book exam. No cheat sheet allowed.Assessment criteria: The exam consists of two parts: mid-term evaluation and final evaluation, each of which counts 25%.Task 3: Written exam – RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: The objective of this final exam is to test the student’s understanding of the course material.Assessment rules: The student has to answer questions with pencil and paper. This is a closed-book exam. No cheat sheet allowed.Assessment criteria: The student must answer the stipulated questions in a way that clearly demonstrates understanding of underlying concepts.Assessment modality: Combined assessmentAssessment tasksAssessment modality: Combined assessmentAssessment tasks

Note

Course materialsSyllabus☐Yes☒NoRemarks:Literature list☐Yes☒NoRemarks:Moodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=321Other, please specify:

INTRODUCTION TO PROJECT MANAGEMENT

Objectives

The course aims to provide comprehensive knowledge and practical skills in project management for scientific research, encompassing the entire lifecycle of projects from initiation to completion, including defining project deliverables, developing project plans, employing best practices and modern tools, conducting research ethically, and effectively presenting research findings.

Course learning outcomes

At the end of the course students should be able to:- Understand Key Concepts of Project Management- Develop Practical Project Management Skills- Apply Best Practices in Project Management- Master Research Development Techniques- Handle and Analyze Data Ethically- Effectively Communicate Research Findings

Description

This course provides an in-depth exploration of project management and scientific research, covering project planning, deliverables, values, tools, best practices, and case studies, as well as scientific research processes from topic selection to data gathering, hypothesis generation, and ethical considerations, with practical hands-on lab sessions integrated throughout each week.

Assessment

Assessment modality: Combined assessmentAssessment tasks

Task 1: Take-home assignment (30%)Grading scheme: 20 points (0-20)Objectives: Written one report for project management planAssessment rules: Match with the course quality criteria.Assessment criteria: Time constrained submission.Task 2: Take-home assignment (30%)Grading scheme: 20 points (0-20)Objectives: Written one report for scientific research.Assessment rules: Match with the course quality criteria.Assessment criteria: Time constrained submission. Task 3: Presentation (40%)Grading scheme: 20 points (0-20)Objectives: Presentation to a group of studentsAssessment rules: Match with the course quality criteria.Assessment criteria: Time constrained submission and present.Task 4: Written exam – RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: In the case the student has failed to pass the class, he/she will be given a written retake assignment.Assessment rules: Match with the course quality criteria.Assessment criteria: Time constrained submission.

Note

Course materials

Syllabus☐Yes☒NoRemarks:Literature list☒Yes☐NoRemarks:The AI Revolution in Project Management: Elevating Productivity with Generative AI Vijay Kanabar (Auteur), Jason Wong (Auteur) ISBN 978-0138297336 Project Management: A Systems Approach to Planning, Scheduling, and Controlling Harold Kerzner ISBN: 978-1119805373 Software Engineering Ian Sommerville ISBN: 978-0133943030 Engineering Software Products: An Introduction to Modern Software Engineering, Global Edition, 1st edition Ian Sommerville ISBN-13: 9781292376356 https://en.wikibooks.org/wiki/LaTeX Moodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=325 Other, please specify:

2nd Semester
THEORETICAL COMPUTER SCIENCE 1
Objectives
This course covers basic principles and techniques in theoretical computer science. More specifically, automata theory, formal languages and Turing machines will be the core part of the course. The course is structured into two main parts. In the first part, we introduce the basic concepts of automata theory such as deterministic and non-deterministic automata, regular and context-free languages. We then move to more advanced concepts of automata theory and computation. The languages and methods covered in the course are the building blocks of the scientific foundation of computer science.

Course learning outcomes
By the end of this course, the students will be able to:1. Understand the hierarchy of regular, context-free and decidable languages.2. Know the formal definition of syntax and semantics of various types of computational models (including (non-)deterministic finite automata, regular expressions, push-down automata, context-free grammars, Chomsky normal form, and Turing machines).3. Understand how the various types of computational models relate to the hierarchy of languages.4. Given an informal or mathematical definition of a formal language, develop a model of a given type that generates that language, and vice versa.5. Given an informal or mathematical definition of a formal language, determine to which language class it belongs and, using one of the pumping lemmas, prove that it does not belong to a language class that is lower in the hierarchy.6. Convert a computational model of a given type into an equivalent computational model of another given type.
Description
1. Basic Automata Theory and Computability 1.1. Automata Theory: introduction, context, motivation, history, notation, basic concepts 1.2. Motivation and history, Deterministic finite automata 1.3. Non-deterministic finite automata, equivalence of deterministic and non-deterministic automata 1.4. Regular expressions, closure properties of regular languages 1.5. equivalence of regular languages and finite automata, non-regular languages 1.6. Pumping lemma for regular languages, Context-free languages, context-free grammars, parsing, ambiguity 1.7. Pushdown automata 2. Advanced Automata Theory and Computability 2.1. Inclusion of regular languages in context-free languages, equivalence of pushdown automata and context-free grammars 2.2. Chomsky normal forms, pumping lemma for context-free languages 2.3. Turing machines 2.4. Turing-decidability, Turing machine variants 2.5. Church-Turing thesis, decidable problems
Assessment
Assessment modality: Combined assessment

Assessment tasks:

Task 1 : Mid-term written exam (50%)Grading scheme: 20 points (0-20)Objectives Exam 1: Evaluate the mastery of contents taught in the first half of the semester. Assessment rules:

Closed exam without any supporting materials only eligible for students who take the course for the first time Assessment criteria: The correctness of solutions to assigned questions Task 2: Written exam (50%)Grading scheme: 20 points (0-20)Objectives Exam 2: Evaluate the mastery of contents taught in the second half of the semester. Assessment rules: Closed exam without any supporting materials only eligible for students who take the course for the first time and pass the mid-term exam Assessment criteria: The correctness of solutions to assigned questions Task 3: Written exam RETAKE (100%)Grading scheme: 20 points (0-20) Objectives Full Exam: Evaluate the mastery of all the contents given in the whole semester. Provide a chance for students who need retaking the exam. Assessment rules: Closed exam without any supporting materials eligible for the students who fail in exams of previous semesters or the mid-term exam in the same semester. The Full Exam is equivalent to the combination of Exam 1 and Exam 2. Assessment criteria: The correctness of solutions to assigned questions
Note
Course materials

Syllabus

☒Yes ☐No

Remarks:

Published on Moodle on a weekly basis

Literature list

☒Yes ☐No

Remarks:

Michael Sipser, Introduction to the Theory of Computation, 3rd Ed. (EMEA adapta- tion), Cengage Learning, 2013

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=2454

Other, please specify:

COMPUTING INFRASTRUCTURES 1

Objectives

Computing professionals should not regard computers as black boxes that execute programs by magic. This course first aims to develop a deeper understanding of the hardware and software environment upon which all computing is based. Students should acquire an understanding of a computer system's functional components, their characteristics, performance, and interactions. The course will cover how data are represented in memory and how they are manipulated during computations. This course provides an introduction to operating systems, which define an abstraction of hardware and manage resource sharing among users. For instance, the course introduces the concept of processes and describe the difference between kernel and user execution modes.

Course learning outcomes

At the end of the course the student will be able to:

•Explain why everything is data, including instructions, in computers.•Describe how positive integers are stored in unsigned representation.•Describe how negative integers are stored in two's-complement representation.•Describe how floating-point numbers are stored in IEEE754 representation.•Describe the internal representation of non-numeric data, such as characters and strings.•Convert numerical data from one format to another.•Explain how fixed-length number representations affect accuracy and precision.•Explain the organization of the classical von Neumann machine and its major functional units.•Describe how an instruction is executed in a classical von Neumann machine.•Understand the mapping between high-level language code and assembly/machine language code.•Explain different instruction formats, such as addresses per instruction and variable length vs. fixed length formats.•Explain the basic concepts of interrupts and I/O operations.•Identify the main types of memory technology (e.g., SRAM and DRAM) and their relative cost and performance.•Explain memory hierarchy and cost-performance trade-offs.•Describe how the use of memory hierarchy (cache, virtual memory) reduces the effective memory latency.•Explain how interrupts are used to implement I/O control and data transfers.•Explain basic instruction level parallelism using pipelining.•Explain the importance of spatial and temporal locality in determining performance.•Explain the objectives and functions of modern operating systems.•Describe how computing resources are used by application software and managed by system software.•Contrast kernel and user mode in an operating system•Describe the need for memory protection in an OS.•Discuss the advantages and disadvantages of using interrupt processing.•Explain the different states that a task may be in•Describe reasons for using interrupts, dispatching, and context switching to support concurrency in an operating system. •Describe the difference between processes and threads.•Explain how processes and threads are created and terminated.•Compare and contrast common algorithms used for pre-emptive and non-pre-emptive scheduling.

Description

Course content Overview and history of computer architectureMachine level data representation: • Bits, bytes, and words• Numeric data representation and number bases: fixed-point, floating-point, signed and two's-complement representations, • Representation of

non-numeric data such as characters• Overflow and underflow of numeric data typesComputer architecture: • Basic building blocks of a computer• von Neumann machine architecture• Control unit, instruction fetch, decode, and execution.• Shared memory multiprocessors/multicore organization• Main memory organization and operations• Cache memories and storage systems• Memory hierarchy: the importance of temporal and spatial locality• Direct-memory access (DMA)• Instruction set architecture (ISA) • Instruction formats• Data manipulation, control, I/O• Addressing modes• Assembly and machine language • Subroutine call and return mechanisms.• I/O and interrupts• Heap, stack, data, and code segments• Branch prediction, speculative execution, out-of-order executionOperating Systems (OS):• Role and purpose of the operating system• OS main abstractions, processes, and resources• Concept of application program interfaces (APIs) • Use of libraries• The evolution of hardware/software techniques and application needs• Device organization• Interrupts: principles and implementations• Concept of user/system state and protection, transition to kernel mode• Process scheduling, scheduling policies and context switching.• The role of interrupts.• The concept of threads versus processes.
Assessment
Assessment modality: Combined assessment

Assessment tasks

Task 1: Written exam (60%)Grading scheme: 20 points (0-20)Objectives: Assess students learning. Assessment rules: Closed-book exam. Assessment criteria: Correct and well-explained answers to the questions. Demonstrate the acquisition of the learning outcomes of the course. No minimum passing grade is required for the exam but the final grade, calculated as 0.6*Task 1 + 0.4*Task 2, must be greater than or equal to 10 to pass the course. Task 2: Take-home assignment (40%)Grading scheme: 20 points (0-20)Objectives: Assess students learning and improve learning. Identify gaps between what students know and can do and what they need to know and be able to do. Assessment rules: Each week students must hand in an assignment. Assignments are submitted electronically. Over the course of semester, four assignments are taken at random to be graded. Those four assignments together weight 40% of the final grade (10% each). Assessment criteria: Correct and well-explained answers to the questions. Demonstrate the acquisition of the concepts presented during the lectures. No minimum passing grade is required for the assignments but the final grade, calculated as 0.6*Task 1 + 0.4*Task 2, must be greater than or equal to 10 to pass the course. Task 3: RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: Students having failed the course will have to re-sit the final written exam at a next exam session. The exam will count for the entirety of the final grade. However, optionally, students may decide to keep the grade obtained for the assignments, which will then count for 40% of the final grade. The minimum passing final grade for the course is 10. Assessment rules: Closed-book exam. Assessment criteria: Correct and well-explained answers to the questions. Demonstrate the acquisition of the learning outcomes of the course. The minimum passing grade is 10/20.
Note
Course materials

Syllabus

☒Yes ☐No

Remarks:

Available on Moodle.

Literature list

☒Yes ☐No

Remarks:

– Structured Computer Organization: International Edition, by Andrew S. Tanenbaum, Todd Austin, Person, 6th edition, 2012.- Modern operating systems, by Andrew S Tanenbaum, Herbert Bos, 4th edition, 2015.- Operating System Concepts, by Abraham Silberschatz et al, 8th edition, 2008.- Computer Systems: a Programmer's Perspective, by Randal E. Bryan, David R. O'Hallaron, 2nd edition, 2011.

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=2453

NETWORKING AND COMMUNICATION

Objectives

Networking and communication play a central role in interconnected computer systems that are transforming the daily lives of billions of people. The Internet provides connectivity for networked applications that serve ever-increasing numbers of individuals and organizations around the world. Many computing applications that are used today would not be possible without networks. The objective of the course is to introduce networks in general, and TCP/IP networks in particular. The course will introduce current network architectures, application-level protocols (e.g., HTTP), transport protocols (TCP/UDP), routing and IP. The course will introduce Medium Access Control (MAC) techniques and quantitative performance measures for networks. The course will also provide a first introduction to mathematical concepts of networks.

Course learning outcomes

By the end of this course, the students will be able to:- understand the role and operation of networking and communication in networked computer systems,- explain how network structure and communication protocols affect the behavior of distributed applications,- describe key networking and communication principles and their specific instantiations on the Internet,- apply the concepts in new technological settings.

Description

1. Introduction to Computer Networks and the Interneta. Importance of networking in contemporary computing, and associated challenges.b. Organization of the Internet (e.g., users, Internet Service Providers, autonomous systems, content providers, content delivery networks).c. Switching techniques (e.g., circuit and packet).d. Layers and their roles (application, transport, network, datalink, and physical).e. Layering principles (e.g., encapsulation and hourglass model).f. Network elements (e.g., routers, switches, hubs, access points, and hosts).g. Basic queueing concepts (e.g., relationship with latency, congestion, service levels, etc.)2. Networked Applicationsa. Naming and address schemes (DNS, IP addresses, and Uniform Resource Identifiers).b. Distributed application paradigms (e.g., client/server, peer-to-peer, cloud, edge, and fog).c. Diversity of networked application demands (e.g., latency, bandwidth, and loss tolerance).d. An explanation of at least one application-layer protocol (e.g., HTTP, SMTP, and POP3).e. Interactions with TCP, UDP, and Socket APIs.3. Reliability Supporta. Unreliable delivery (e.g., UDP).b. Principles of reliability (e.g., delivery without loss, duplication, or out of order).c. Error control (e.g., retransmission, error correction).d. Flow control (e.g., stop and wait, window based).e. Congestion control (e.g., implicit, and explicit congestion notification).f. TCP and performance issues (e.g., Tahoe, Reno, Vegas, Cubic, QUIC).4. Routing and Forwardinga. Routing paradigms and hierarchy (e.g., intra/inter domain, centralized and decentralized, source routing, virtual circuits, QoS).b. Forwarding methods (e.g., forwarding tables and matching algorithms).c. IP and Scalability issues (e.g., NAT, CIDR, BGP, different versions of IP).5. Single-Hop Communicationa. Introduction to error detection and correction techniques.b. Medium Access Control (MAC) (e.g., random access and scheduled access).c. Ethernet.d. Switching.e. Link virtualization.

Assessment

Assessment modality:

Type of assessment

Assessment tasks

Task 1: Written exam – Midterm (50%)Grading scheme: 20 points (0-20)Objectives: Assess students' knowledge, skills, and abilities. Assessment rules: The students will have 45 minutes to complete the written exam. Only clearly crossed answers will be marked as correct. In case of doubts, the side calculations will also be taken into consideration. Books, lecture notes, and calculators are allowed to be used during the exam. No communication device whatsoever is allowed during the exam (e.g., laptop, smartphone, smartwatch, etc.).Assessment criteria: A number of points will be assigned for each correct answer. The number of points for each correct answer will depend on the difficulty of the question.Task 2: Written exam (50%)Grading scheme: 20 points (0-20)Objectives: Assess students' knowledge, skills, and abilities. Assessment rules: The students will have 45 minutes to complete the written exam. Only clearly crossed answers will be marked as correct. In case of doubts, the side calculations will also be taken into consideration. Books, lecture notes, and calculators are allowed to be used during the exam. No communication device whatsoever is allowed during the exam (e.g., laptop, smartphone, smartwatch, etc.).Assessment criteria: A number of points will be assigned for each correct answer. The number of points for each correct answer will depend on the difficulty of the question.Task 3: Take-home assignmentGrading scheme:Pass/Fail Up to 20% bonus (4 points)Objectives: Understand network protocols in action by observing the sequence of messages exchanges between two protocol entities, delving down into the details of protocol operation, and causing protocols to perform certain actions and then observing these actions and their consequences.Assessment rules: Take-home assignments will be solved by groups of at least 2 people/group. At least 4 different assignments will be given during the semester. Take-home assignments are not mandatory. Each passed assignment will give the student a 5% bonus (1 point), up to a maximum of 20% (4 points). The bonus will be added to the final grade of the written exam only if the student obtained the minimum passing grade (i.e., 10 points).Assessment criteria: Each submitted assignment will receive a Pass/Fail assessment based on its correctness.Task 4: RETAKEGrading scheme: 20 points (0-20)Objectives: Students having failed the course will have to re-sit the written exam at a next exam session. Any bonus previously obtained will remain valid and will be added to the final grade of the written exam only if the student obtained the minimum passing grade (i.e., 10 points).Assessment rules: Same as Task 1 and 2Assessment criteria: Same as Task 1 and 2
Note
Syllabus

☒Yes ☐No

Remarks:

Available on Moodle.

Literature list

☒Yes ☐No

Remarks:

Computer Networking: A Top Down Approach 8th edition (Pearson) – available at the LLC.

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=2455

Other, please specify:

LINEAR ALGEBRA 2

Objectives
Fundamental notions of Linear Algebra

Course learning outcomes
The students recall and are able:

To formulate the basic notions of linear algebra.

To solve small linear algebra problems by performing calculations and applying algorithms.

To understand small proofs and to look for examples and counterexamples.
Description
General vector spaces (base change); matrix spaces, matrix transformations; Eigenvalues and Eigenvectors; complex vector spaces ; inner product spaces, orthogonality; quadratic forms; general linear transformations; if time permits, some applications of linear algebra.
Assessment
Assessment modality: Combined assessment

Assessment tasks

Task 1: Written exam – Midterm exam 1 (15%)Grading scheme: 20 points (0-20)Objectives: Written test with exercises.Assessment rules: Only 6 pages of personal notes are allowed. No calculators.Assessment criteria: The tests are graded. The average of the two best midterms counts as mark.Task 2: Written exam – Midterm exam 2 (15%)Grading scheme: 20 points (0-20)Objectives: Written test with exercises.Assessment rules: Only 6 pages of personal notes are allowed. No calculators.Assessment criteria: The tests are graded. The average of the two best midterms counts as mark.Task 3: Written exam – Midterm exam 3 (15%)Grading scheme: 20 points (0-20)Objectives: Written test with exercises.Assessment rules: Only 6 pages of personal notes are allowed. No calculators.Assessment criteria: The tests are graded. The average of the two best midterms counts as mark.Task 4: Written exam – Final exam (70%)Grading scheme: 20 points (0-20)Objectives: Written test with exercises.Assessment rules: Only 6 pages of personal notes are allowed. No calculators.Assessment criteria: The test is graded.Task 5: Oral exam, optionalGrading scheme: 20 points (0-20), Objectives: Oral exam (in case of justified absence to >1 midterms or as additional evaluation in case of fraud suspicion)Assessment rules: Discussion with questions (theoretical questions and exercises)Assessment criteria: Correct and partially correct answers contribute positively to the grade.Task 6: Either final written exam (100%) or the same modalities as the regular students (to be selected by each retaking student at the beginning of each semester)Grading scheme: 20 points (0-20), Objectives: Written test with exercises. In case of fraud suspicion, the exam can be replaced by an oral exam (as described in Task 5). Assessment rules: Only 6 pages of personal notes are allowed. No calculators.Assessment criteria: The test is graded.
Note

Syllabus

☒Yes ☐No

Remarks:

Moodle page

Literature list

☒Yes ☐No

Remarks:

Moodle page

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=2456

Other, please specify:

PROGRAMMING FUNDAMENTALS 2

Objectives

This course is about object-oriented programming, a paradigm that enables programmers to deal with program complexity by decomposing programs into small, self-contained units that can easily be reused and adapted across projects. The course will have a practical flavour, with demonstrations, examples, and assignments. We will rely on the Java programming language to concretize object-oriented concepts through the development of programs. Java is among the five most popular and most demanded programming languages on the job market; it is used in Web-based, Android, and embedded systems. Java is popular because it is easy to learn, has a rich set of programming APIs, and is supported by many development tools. In this course, through Java, the students will learn to develop software applications of medium complexity relying on class inheritance and decomposition, known Java data structures, exception handling, file processing, GUIs, and concurrency support.

Course learning outcomes

The course will lead to the following learning outcomes:Design, code, test, and debug Java programs that follow an object-oriented design and uses each of the following fundamental programming constructs: classes, assignment and expressions, console I/O, conditional and iterative structures, functions with parameter passing, structured data types provided with the language, use file I/O to provide persistence across multiple executions, rely on exception-handling mechanism.Write programs of medium complexity that use Java GUI APIs and rely on concurrency mechanisms.Develop tests for program modules and apply a variety of strategies to design test cases.Build, execute and debug programs using a modern IDE and associated tools such as visual debuggers.

Description

The course has the following lesson topics:1. Development environments: shell, editor, java compiler vs runtime, source code control, build automation tools, IDEs,2. Java Basics: Types, Control Flow, and I/O operations,3. Inheritance and Polymorphism in Java,4. Implementing data structures in Java,5. Java collections,6. Parametric polymorphism and generics,7. Exception handling in Java,8. File processing,9. Concurrency in Java,10. Java GUIs, 11. Event-driven programming in Java,12. Static methods, Nested classes, Networking,13. Persistence,14. Lambda and Streams.They cover for the following teaching objectives:A) Principles of object-oriented programming and designA.1) Decomposition into objects carrying state and having behaviour through the definition of classes (fields, methods, and constructors), subclasses, inheritance, and method overriding.A.2) Idioms for encapsulation (visibility, interfaces, and abstract classes).A.3) Dynamic dispatching of method calls definition of method-call.B) The Java language as an example of object-oriented languageB.1) Basic concepts such as variables, primitive data types, expression evaluation, assignment. B.2) Basic constructs such as conditional and iterative structures and flow of control. B.3) Key modularity constructs such as methods and classes, and related concepts like parameter passing, scope, abstraction, data encapsulation.B.4) Input and output using files, console, and APIs B.5) Structured data types available in the Java APIs (e.g., the collection framework) B.6) GUI Libraries B.7) RecursionB.8) Dealing with runtime errors in programs (exception handling) B.9) Strings and string processingC) Data structures in JavaC.1) Implementing standard abstract

data types such as lists and trees in JavaC.2) The Java Collections package for lists, trees, stacks, queues, sets, and mapsC.3) Performance implications of choice of data structure(s)D) Principles of reactive programmingD.1) Components of reactive programming: event-source, event signals, listeners and dispatchers, event objects, adapters, event-handlers.D.2) Use of reactive programming in Java, with a GUI case study: Defining event handlers/listeners, Parameterization of event senders and event arguments, externally generated events, and program-generated events D.3) Conceptual separation between Model, View, and Controller.E) Parallelism and concurrency in JavaE.1) Basic constructs and the concurrent Package.F) Basic software testing principlesF.1) Deriving test cases from functional specifications and implementationF.2) The Junit frameworkF.3) Code coverageG) Development environmentsG.1) Shells, editors, java compiler vs runtime.G.2) Source code control, build automation tools, IDEs.
Assessment
Assessment modality: Combined assessmentAssessment tasksTask 1: Mid-term exam (30%) Grading scheme: 20 points (0-20)Objectives: Written exam to assess students' knowledge of the basics of Java and OO programming. Consists of short programming exercises, open questions, and quizzes. Assessment rules: All enrolled students. Laptop, smartphones, cheat sheets, and books are not allowed. Assessment criteria: Correctness of answers, program design, functioning, and code style. Task 2: Written exam (70%)Grading scheme: 20 points (0-20)Objectives: Programming exercises to assess that students have reached the objectives of the course. Assessment rules: All enrolled students. Laptop, smartphones, cheat sheets, and books are not allowed. Assessment criteria: Correctness of program design, functioning, and code style. Task 3: RETAKEGrading scheme: 20 points (0-20)Objectives: Programming exercises to assess that students have reached the objectives of the course. The result of mid-term exam is not considered for the final grade. Assessment rules: All enrolled students. Laptop, smartphones, cheat sheets, and books are not allowed. Assessment criteria: Correctness of program design, functioning, and code style.
Note
Course materialsSyllabus ☒Yes☐No Remarks: Available on Moodle. Literature list ☒Yes☐No Remarks: 1) Building Java Programs, 4th Edition. Stuart Reges, Marty Stepp. Pearson. 2) Learning Java. Marc Loy, Patrick Niemeyer, Daniel Leuck. 6th Edition. O'Reilly Media. Moodle page ☒Yes☐No Remarks: https://moodle.uni.lu/course/view.php?id=2452 Other, please specify:

BACHELOR SEMESTER PROJECT 2

Objectives

During a standard BSP, students discover research and development domains, produce concrete artefacts related to computer science knowledge areas covered in the BICS, collaborate with their academic advisor(s), learn new technologies related to computer science, learn new knowledge related to computer science, apply the scientific and technical knowledge learned during the BICS, apply the primary and secondary languages knowledge learned during the BICS. The course content and learning objectives are consistent with the BiCS programme content that the student is following.

Course learning outcomes

At the end of the BSP, the student should have acquired specific learning outcomes defined on an individual basis between the academic advisor(s) and him/herself. The learning outcomes should be agreed upon at the beginning of each project and be aligned with the general BiCS curriculum.

Description

Specific course content is defined on an individual basis between the academic advisor(s) and the student for each project.

Assessment

Assessment modality: Combined assessment

Task 1: Final report and in-person presentation (100%)

Grading scheme: 20 points (0-20)Objectives: Assess the quality and quantity of work done during the semester.Assessment rules: Time constrained submissions.Assessment criteria: The detailed criteria are specific to each project subject and applied on the following weighted deliverables:•technical & scientific quality of the final report, primary language (60%)•technical & scientific quality of the final in-person presentation, primary language (30%)•technical & scientific quality of the final report, secondary language summary (5%)•technical & scientific quality of the final in-person presentation, secondary language summary (5%)Task 2: Retake (100%)Grading scheme: 20 points (0-20)Objectives: Option to repeat a previously failed project (with a final assessment of <10 points) with the same topic and under the same supervisor.Assessment rules: The same deliverables as for the previously failed project must be resubmitted under the given time constraints.Assessment criteria: Same as for Task 1.

Note

Syllabus☒Yes☐NoRemarks:Accessible from the Moodle course pageLiterature list☒Yes☐NoRemarks:Accessible from the Moodle course pageMoodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=2457

3rd Semester

STATISTICS FOR COMPUTER SCIENTISTS

Objectives

Statistical methods are central in data science, machine learning, and computer science at large as they provide a sound framework for understanding and making sense of the data. In this course whose orientation is practical, students will learn the foundation to choose and apply the proper statistical techniques to solve a given problem. Practice exercises will develop proficiency in the Python programming language and in data analysis in general.

Course learning outcomes

At the end of the course the student will be able to:•Understand the basic concepts: population, sample, measures of dispersion, variance.•Display data graphically and interpret graphs (e.g., histograms).•Recognize, describe, and calculate means, medians, quantiles, variances, interquartile ranges.•Compute confidence intervals with formulas and bootstrapping.•Understand why random sampling can reduce bias and yield a higher quality data set.•Formulate and understand the concept of data correlation.•Data transformation: data cleaning and dimension reduction.•Apply Principal Component Analysis (PCA) to dimensionality reduction problems.•Apply linear regression to curve-fitting problems.•Understand the concept of overfitting.•Discuss and apply cross-validation in the context of overfitting and model selection (e.g., degree of polynomials in a regression context).•Apply statistical tests to compare different design candidates.•Understand how statistics can be used to set engineering simulation parameters with respect to target confidence level.•Apply statistics to the analysis of financial data (e.g., correlation between markets).•Demonstrate proficiency in Python.

Description

The course covers the following topics:• Overview of the use of statistics in data analysis, machine learning and engineering. • Descriptive statistics and exploratory data analysis: population, sample, mean, median, variance, empirical distribution histogram, cumulative distribution function, quantile, box plots, density plots, scatterplot and coefficient of correlation.• Inferential statistics: sampling, sampling bias and bootstrap.• Confidence intervals and margin of errors.• A/B testing.• Statistical significance and p-values.• Hypothesis tests: one way versus two ways tests.• Performance comparisons, evaluating experiments.• Statistically-sound simulation-based design in electrical engineering.• Basic statistical methods used in financial data analysis.• Statistics in the performance evaluation of safety-critical computer systems.

Assessment

Assessment modality: Combined assessmentAssessment tasksTask 1: Written exam (60%)Grading scheme: 20 points (0-20)Objectives: Assess students learning.Assessment rules: Closed-book exam. Assessment criteria: Correct and well-explained answers to the questions. Demonstrate the acquisition of the learning outcomes of the course. No minimum passing grade is required for the exam but the final grade, calculated as 0.6*Task 1 + 0.4*Task 2, must be greater than or equal to 10 to pass the course.Task 2: Take-home assignment (40%)Grading scheme: 20 points (0-20)Objectives: Assess students learning and improve learning. Identify gaps between what students know and can do, and what they need to know.Assessment rules: Group work (2 or 3 students per group).Assessment criteria: Correct and well-explained answers to the questions. The quality of the report (writing, presentation) is an important

assessment criterion. Submissions will be systematically checked for potential plagiarism. Task 3: RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: Students having failed the course will have to re-sit the final written exam at a next exam session. The exam will count for the entirety of the final grade. However, optionally, students may decide to keep the grade obtained for the assignments, which will then count for 40% of the final grade. The minimum passing final grade for the course is 10.Assessment rules: Closed-book exam.Assessment criteria: Correct and well-explained answers to the questions. Demonstrate the acquisition of the learning outcomes of the course. The minimum passing grade is 10/20.
Note
Course materials

Syllabus☒Yes☐NoRemarks:Available on MoodleLiterature list☒Yes☐NoRemarks:-“Practical Statistics for Data Scientists”, P. Bruce, A. Bruce, P.Gedeck, second edition, O’Reilly, 2020.-“Performance Evaluation Of Computer And Communication Systems”, J.Y. Le Boudec, EPFL press, 2010. Available at https://leboudec.github.io/perfeval/ Moodle page☒Yes☐NoRemarks:TBC

DISCRETE MATHEMATICS 2

Objectives

Discrete structures are foundational material for computer science. Relatively few computer scientists will be working primarily on discrete structures, but many other areas of computer science require the ability to work with concepts from discrete structures. Second part of the course focuses on the data analysis. Methods used come from combinatorics, probability theory and statistics

Course learning outcomes

At the end of the course the student will be able to:-Find cardinalities of discrete structures.-Use counting for proofs.-Compute probabilities of events based on observations, and independence.-Take decisions via probabilities.

Description

1. Discrete Structures (DS)1.1. Basics of Counting1.1.1. Counting arguments: Set cardinality and counting, Sum and product rule, Inclusion-exclusion principle.1.1.2. The pigeonhole principle1.2. Permutations and combinations, Pascal's identity, The binomial theorem1.3. Solving recurrence relations (cross-reference: AL/Basic Analysis): An example of a simple recurrence relation, such as Fibonacci numbers.2. Discrete Probability2.1. Finite probability space, events2.2. Axioms of probability and probability measures2.3. Conditional probability, Bayes' theorem2.4. (Conditional) Independence2.5. Expectation and its Properties, Mean and Variance2.6. Integer Random Variables (Bernoulli, Binomial, etc..)2.7. Continuous Random Variables (Gauss, Poisson, etc..)2.8. Law of Large Numbers, Central Limit Theorem

Assessment

Assessment modality: Combined assessmentAssessment tasks

Task 1: Written exam (35%)Grading scheme: 20 points (0-20)Objectives: Check if the students are capable of formalisation, and computation of counting problems. Furthermore, algebraic methods are expected to be known by then. Assessment rules: One A4 handwritten cheat sheet. Possibly needed tables are provided. The exam is based on both questions and longer problems.Assessment criteria: Correctness, and chosen path to a solution.Task 2: Written exam (65%)Grading scheme: 20 points (0-20)Objectives: Final exam counting and ProbabilitiesAssessment rules: One A4 handwritten cheat sheet. Possibly needed tables are provided. The exam is based on both questions and longer problems.Assessment criteria: Correctness, and chosen path to a solutionTask 3: Oral exam – RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: Retake students will undergo evaluation to determine their proficiency in formalization and computation of counting problems, as well as their grasp of fundamental concepts in probability.Assessment rules: Oral examAssessment criteria: 75% CPE 1 and 25% CPE 2

Note

Syllabus☒Yes☐NoRemarks:Available on the Moodle page.Literature list☒Yes☐NoRemarks:Based on the lecture notes from MIT, see https://ocw.mit.edu/courses/6-042j-mathematics-for-computer-science-fall-2005/pages/lecture-notes/ Moodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=3329 Other, please specify:All the lecture slides and old videos are shared in Moodle page

INFORMATION MANAGEMENT 1

Objectives

Information Management is primarily concerned with the capture, digitization, representation, organization, transformation, and presentation of information; algorithms for efficient and effective access and updating of stored information; data modelling and abstraction; and physical file storage techniques. The student needs to be able to develop conceptual and physical data models, determine which IM methods and techniques are appropriate for a given problem, and be able to select and implement an appropriate IM solution that addresses relevant design concerns including scalability, accessibility, and usability.

Course learning outcomes

At the end of the course, students will be able to:-design data models in different representations (object-oriented, hierarchical, network-based, relational)-avoid redundancies and inconsistencies in the resulting models.-use different query languages to create and query large databases.-solve and implement individual information-management use-cases.

Description

Course content 1. Information Management (IM) 1.1. Information Management Concepts 1.1.1. Information systems as socio-technical systems 1.1.2. Basic information storage and retrieval (IS&R) concepts. 1.1.3. Information capture and representation 1.1.4. Supporting human needs: searching, retrieving, linking, browsing, navigating1.1.5. Information management applications 1.1.6. Declarative and navigational queries, use of links 1.1.7. Content analysis and indexing 1.1.8. Quality issues: reliability, scalability, efficiency, and effectiveness 1.2. Relational Databases 1.2.1. Mapping conceptual schema to a relational schema 1.2.2. Keys and foreign-keys, referential integrity 1.2.3. Relational algebra and relational calculus 1.2.4. Relational database design 1.2.5. Functional dependencies 1.2.6. Decomposition of a schema; lossless-join and dependency-preservation properties of a decomposition. 1.2.7. Candidate keys, super keys, and closure of a set of attributes 1.2.8. Normal forms (2NF, 3NF BCNF) 1.2.9. Multi-valued dependencies (4NF) 1.2.10. Join dependencies (PJNF, 5NF) 1.2.11. Representation theory 1.3. Query Languages 1.3.1. Overview of database languages 1.3.2. SQL (data definition, query formulation, update sublanguage, constraints, integrity) 1.3.3. Select-project-join queries. 1.3.4. Aggregations and group-by 1.3.5. Over-operator and sliding windows. 1.3.6. Subqueries in SQL 1.3.7. Constraints and triggers 1.3.8. Stored procedures and PL/SQL1.3.9. QBE and 4th-generation environments 1.3.10. Different ways to invoke non-procedural queries in conventional languages. 1.3.11. Overview of other major query languages (e.g., XPATH, SPARQL)

Assessment

Assessment modality: Combined assessmentAssessment tasksTask 1: Written or oral midterm exam (50%)Grading scheme: 20 points (0-20)Objectives: Reproduction and application of the course contentsAssessment rules: Closed book written exam (whole class) or 30 minutes individual oral exam.Assessment criteria: Exam results are based on the correctness of the provided answers.Task 2: Written or oral final exam (50%)Grading scheme: 20 points (0-20)Objectives: Reproduction and application of the course contentsAssessment rules: Closed book written exam (whole class) or 30 minutes individual oral exam.Assessment criteria: Exam results are based on the correctness of the provided answers.Task 3: Active participation –

Presentation of exercise solutionsUp to 2 bonus pointsGrading scheme: 20 points (0-20)Objectives: Solution of exercise problemsAssessment rules: A correct presentation (in class) of the solution to one problem of an exercise sheet yields 1 bonus point. Up to 2 bonus points can be achieved per student this way per semester. Bonus points are added on top of the points the student has achieved through the exams. At most 20 points can be achieved in altogether.Assessment criteria: Correctness of the provided solutions.Task 4: Written exam – RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: Reproduction and application of the course contentsAssessment rules: Closed book written exam (whole class) or 30 minutes individual oral exam.Assessment criteria: Solution of exercise problems
Note
Course materialsSyllabus☒Yes☐NoRemarks:Available on MoodleLiterature list☒Yes☐NoRemarks:Database Systems: The Complete Book (ISBN: 978-0131873254)Moodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=3332

PROGRAMMING FUNDAMENTALS 3

Objectives

Programming fundamentals 3 is a two parts course where you will learn functional programming and parallel programming. The first part focusses on the foundation of the functional language OCaml. We show that functional programs are often shorter and more elegant than their imperative counterparts. In the second part, we introduce concurrent and parallel programming. We discuss general synchronisation issues of concurrency models such as deadlock, livelock and starvation. We present the shared state and message-passing programming models aiming at solving these issues. Moreover, we connect both paradigms as the full class will be in the newest OCaml 5.0 language, which includes a new multicore extension.

Course learning outcomes

At the end of the course the student will:-Know how to program and reason about programs in a purely functional programming style. -Solve problems in Haskell in a concise, yet expressive, manner.-Decompose complex problems into smaller, manageable subproblems.-Think recursively, make use of higher-order functions, understand the benefits of type systems, and learn how some programming languages like Haskell avoid unnecessary computations via lazy evaluation.-Know common data structures (like stacks, queues, trees, self-balancing trees, and graphs) and how these can be implemented in a functional language with data immutability. -Know how to analyse the run-time complexity of operations on these data structures. -Understand advanced concepts and abstractions of functional programming such as functors, applicative functors, and monads. Understand how to approach and solve classic computer science problems in Haskell with essential techniques such as recursion, permutation generation, brute-force and binary search.

Description

1. Introduction to FP and Haskell 1.1. Programming languages paradigms 1.2. Basic Haskell syntax 1.3. Polymorphism and typeclasses 1.3.1. Types, type inference and polymorphism 1.3.2. Typeclasses 1.4. Recursion 1.4.1. Thinking recursively 1.4.2. Sorting algorithms 1.5. Functions 1.5.1. Pointfree style and infix functions 1.5.2. Composition 1.5.3. Lambda expressions 1.6. Higher order functions 1.6.1. Maps and filters 1.6.2. Folding 1.6.3. Currying and partial function application 1.7. Data types 1.8. Input/Output 1.9. Functional styles in other programming languages 2. Purely functional data structures 2.1. Data immutability 2.2. Trees 2.2.1. Binary search tree2.2.2. Self-balancing tree (AVL) 2.3. Lazy evaluation of infinite data structures 2.4. Graphs and common graph algorithms in FP3. Arithmetic problems 4. Advanced concepts in FP 4.1. Functors 4.2. Applicative functors 4.3. Monads 4.3.1. Maybe, Either 4.3.2. IO 4.3.3. State 5. Solving classic computer science problems, using Haskell

Assessment

Assessment modality: Combined assessmentAssessment tasksTask 1: Take-home assignment (10%)Grading scheme: 20 points (0-20)Objectives: Develop basic algorithms that avoid assigning to mutable state using recursion.Assessment rules: Homework assignments are required to be submitted individually through Moodle. Each student must complete the homework assignments on their own.Assessment criteria: Successfully solves the stated problem with computational efficiency. Code demonstrates conciseness, clarity, and adherence to a functional programming style.Task 2: Take-home assignment (10%)Grading scheme: 20

points (0-20)Objectives: Solve classic computer science problems with compositional reasoning and higher order functions.Assessment rules: Homework assignments are required to be submitted individually through Moodle. Each student must complete the homework assignments on their own.Assessment criteria: Successfully solves the stated problem with computational efficiency. Code demonstrates conciseness, clarity, and adherence to a functional programming style.Task 3: Take-home assignment (10%)Grading scheme: 20 points (0-20)Objectives: Understand AVL trees and analyze the worst-case time complexity of each operation.Assessment rules: Homework assignments are required to be submitted individually through Moodle. Each student must complete the homework assignments on their own.Assessment criteria: Successfully solves the stated problem with computational efficiency. Code demonstrates conciseness, clarity, and adherence to a functional programming style. Additionally, provides analysis of the run-time complexity of operations.Task 4: Take-home assignment (10%)Grading scheme: 20 points (0-20)Objectives: Understand and apply abstract concepts of FP, namely monadsAssessment rules: Homework assignments are required to be submitted individually through Moodle. Each student must complete the homework assignments on their own.Assessment criteria: Successfully solves the stated problem with computational efficiency. Code demonstrates conciseness, clarity, and adherence to a functional programming style.Task 5: Written exam (60%)Grading scheme: 20 points (0-20)Objectives: Evaluate the students' comprehension of the course's material.Assessment rules: Students are permitted to bring a single "cheat sheet" consisting of an A4-sized paper, on which they can include any necessary notes. A minimum grade of 8/20 in the final written exam is required to pass the course.Assessment criteria: Each question has a stipulated mark clearly indicated on the exam.Task 6: Written exam – RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: Students having failed the course may re-sit the final exam at a next exam session. The objective is to reassess the students' comprehension of the course's material.Assessment rules: Students are permitted to bring a single "cheat sheet" consisting of an A4-sized paper, on which they can include any necessary notes. For re-sits, the exam constitutes 100% of the final grade.Assessment criteria: Each question has a stipulated mark clearly indicated on the exam.
Note
Course materials

Syllabus☐Yes☒NoRemarks:Literature list☒Yes☐NoRemarks:[1] Programming in Haskell, 2nd Edition, by Graham Hutton. September 2016, Cambridge University Press, ISBN-13 978-1316626221.[2] Learn You a Haskell for Great Good!, by Miran Lipovaca. April 2011, No Starch Press, ISBN-13 978-1593272838.Moodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=3330

ALGORITHMS AND COMPLEXITY

Objectives

This course presents fundamental algorithms and data structures that are required to solve common problems. The notion of computational complexity of algorithms is introduced, and mathematical techniques are presented to analyse the complexity of the algorithms presented in the course. Finally, an introduction to problem complexity is given.

Course learning outcomes

Upon completion of this course the student should be able to:- design and analyse an algorithm for a given problem- evaluate the computational complexity of an algorithm- reason about the correctness of an algorithm- classify an algorithm according to the basic approach it uses

Description

– Algorithms, and complexity, upper and lower bounds- Elementary data structures: lists, stacks, queue, sets- Advanced data structures: trees and graphs- Sorting- Searching- Hashing – Trees, and algorithm on Trees(binary search trees and (quasi)-balanced trees)- Graphs, and algorithms on Graphs(depth-first search Prim's and Dijkstra's algorithms)9. Complexity theory: P, NP, NP completeness10. (if time) Algorithm on secondary memory

Assessment

Assessment modality: Combined assessmentAssessment tasksTask 1: Written final exam (50%)Grading scheme: 20 points (0-20)Objectives: Test the student's understanding of the course materialAssessment rules: The student has to answer questions with pencil and paper. This is a closed-book exam. No cheat sheet allowed.Assessment criteria: The student must answer the stipulated questions in a way that clearly demonstrates understanding of underlying concepts.Task 2: Active participation – 6 Quizzes in Class (30%)Grading scheme: 20 points (0-20)Objectives: To track and test the student's understanding for each topic.Assessment rules: The student must answer questions with pen/pencil. Each quiz will be 10 minutes. No cheat sheet allowed.Assessment criteria: The student must answer the stipulated questions in a way that clearly demonstrates understanding of underlying conceptsTask 3: Take-home assignment – 2 Assignments (20%)Grading scheme: 20 points (0-20)Objectives: Check the abilities of the students in analytic thinking and in group collaborationAssessment rules: The assignments are group based. For each assignment there will be a weekly track where each member must write his/her related task in this worksheet. Assessment criteria: Students must be separated into several groups. Evaluation will be individual.Task 4: Written exam – RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: Test the student's understanding of the course materialAssessment rules: The student has to answer questions with pencil and paper. This is a closed-book exam. No cheat sheet allowed.Assessment criteria: The student must answer the stipulated questions in a way that clearly demonstrates understanding of underlying concepts.

Note

Course materials

Syllabus☒Yes☐NoRemarks:Available on Moodle.Literature list☐Yes☒NoRemarks:Moodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=3331

BACHELOR SEMESTER PROJECT 3

Objectives

During a standard BSP, students discover research and development domains, produce concrete artefacts related to computer science knowledge areas covered in the BICS, collaborate with their academic advisor(s), learn new technologies related to computer science, learn new knowledge related to computer science, apply the scientific and technical knowledge learned during the BICS, apply the primary and secondary languages knowledge learned during the BICS. The course content and learning objectives are consistent with the BiCS programme content that the student is following.

Course learning outcomes

At the end of the BSP, the student should have acquired specific learning outcomes defined on an individual basis between the academic advisor(s) and him/herself. The learning outcomes should be agreed upon at the beginning of each project and be aligned with the general BiCS curriculum.

Description

Specific course content is defined on an individual basis between the academic advisor(s) and the student for each project.

Assessment

Assessment modality: Combined assessmentAssessment tasksTask 1: Final report and in-person presentation (100%)Grading scheme: 20 points (0-20)Objectives: Assess the quality and quantity of work done during the semester.Assessment rules: Time constrained submissions.Assessment criteria: The detailed criteria are specific to each project subject and applied on the following weighted deliverables:•technical & scientific quality of the final report, primary language (60%)•technical & scientific quality of the final in-person presentation, primary language (30%)•technical & scientific quality of the final report, secondary language summary (5%)•technical & scientific quality of the final in-person presentation, secondary language summary (5%)Task 2: RETAKE 20 points (100%)Grading scheme: 20 points (0-20)Objectives: Option to repeat a previously failed project (with a final assessment of <10 points) with the same topic and under the same supervisor.Assessment rules: The same deliverables as for the previously failed project must be resubmitted under the given time constraints.Assessment criteria: Same as for Task 1.

Note

Course materials

Syllabus☒Yes☐NoRemarks:Accessible from the Moodle course pageLiterature list☒Yes☐NoRemarks:Accessible from the Moodle course pageMoodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=3334

4th Semester
PRACTICAL FUNCTIONAL PROGRAMMING
Objectives
Haskell is a modern programming language with a rich type system and many advanced features that allow efficient, fast and reliable solutions of extremely complex programming tasks. Haskell is currently being increasingly adopted by industry and open-source programmers, with applications ranging document conversion, cryptography and cryptocurrency protocols, hardware design, web services and microservices, proof management and high-reliability computing, and compiler construction. The main aim of the course is to utilize the functional programming fundamentals from the previous courses in a practical environment, showing how to utilize the advanced concepts to vastly simplify practical programming tasks. The course specifically focuses on utilizing typed functional programming constructions for solving tasks that are extremely laborious and error-prone with the current mainstream programming practice. The course consists mainly of lectures, which are supplemented by individual work of the students on the 3 homework assignments.

Course learning outcomes
At the end of the course the student will be able to:

Become productive with Haskell, using it to fulfil day-to-day programming tasks,

Use functional programming in Haskell for industry-related use-cases including systems programming, network and web applications, and game programming,

Understand the practical importance, implementation, and main limitations of the type-inference in programming languages, and

Utilize advanced functional concepts in their programs and in other languages (such as Scala and PureScript), including monads and monad transformers, lenses, PEG parsers and AST processing techniques.
Description
The following topics are covered in the course: Overview of Haskell syntax and semantics Overloading via type classes and the use of most common type classes (Monoid, FAM, container classes, number hierarchy, …) Use of functions as data objects, simulating global state using functions and monads Parsec-style parsing combinators Haskell infrastructure – cabal, Prelude Container libraries in Haskell, and a brief overview of functional data structures Usual compilation pipeline of lazy functional programs, a brief overview of Haskell type system Curry-Howard correspondence Monad transformers Functional references and optics Haskell libraries for text processing Systems programming interface of Haskell (IO), concurrent and parallel programming Web applications (backend applications and API endpoints, middleware, HTML generators, Front-end programming) Processing and generating graphics with Haskell Selection of debugging, testing and benchmarking libraries; property testing Tools for implementing extensible data structures: Higher-order abstract syntax & higher-kinded data,

tagless & final interpreters, free monads, Selective functors, “Scrap Your Boilerplate” approach and generics.
Assessment
Assessment modality: Combined assessmentAssessment tasks

Task 1: Take-home assignment (40%)Grading scheme: 20 points (0-20)Objectives: 3 homework assignments will be given for individual implementation by students. The assignments will provide practical grounds for exercising the material from the lectures, and the tasks will serve as a common base for the exam questions. Assessment rules: The students submit their solutions via Moodle within the specified deadline, following the submission specification given at the lectures and in Moodle. Submissions are expected to be formatted as a Cabal package. Submissions that are mangled, unreadable, delayed beyond the deadline, or visibly do not adhere to submission criteria will not be evaluated. In case the authorship of the solutions is questionable (e.g., solution seems to be copied from other students or generated from language models), all affected submitted solutions will be scored with 0 points. Assessment criteria: The solutions will be evaluated on delivering the required functionality and suitable programming style (terseness, use of the recommended language features). Exact required goals will be listed in Moodle. Task 2: Written exam (60%)Grading scheme: 20 points (0-20)Objectives: Assessing the understanding of the course matter by students by solving several very simple programming tasks. Assessment rules: The student solves several simple tasks on paper. The tasks may include 1) practical use of the Haskell type system 2) finding problems or improvement opportunities in existing Haskell code using the knowledge of Haskell libraries 3) writing short code snippets that solve given simple tasks. All tasks will be selected from the tasks that the students should have already seen or solved during their work on the homework assignments. The students may bring and use any resource printed on paper, including prepared cheatsheets, solutions of the homework assignments, and Haskell programming books, but no communication or use of any kind of computer is permitted Assessment criteria: The provided solutions of the tasks must be correct. Problems in answers that typically occur when programming on paper (such as small problems with code types, easily fixable syntax issues, or exact library function names) will not have impact on the evaluation unless if present in amounts that hamper the assessment of correctness. Task 3: RETAKE (60%)Grading scheme: 20 points (0-20)Objectives: A student who has failed the course is allowed to sit to the next exam call. The objectives are the same as Task 2. The mark the student got from the assessment Task 1 is kept. Assessment rules: Idem to Task 2. Assessment criteria: Idem to Task 2.
Note
Course materials Syllabus ☒Yes☐No Remarks: Available on Moodle. Literature list ☒Yes☐No Remarks: Learn You A Haskell For Great Good (updated version, https://learnyouahaskell.github.io/) Real World Haskell (updated version, https://github.com/tssm/up-to-date-real-world-haskell) Haskell Wikibook (https://en.wikibooks.org/wiki/Haskell) What I Wish I Knew When Learning Haskell (https://github.com/sdiehl/wiwinwlh) Moodle page ☒Yes☐No Remarks: TBD

INFORMATION MANAGEMENT 2

Objectives

This course is a follow up, of IM1. Therefore, the overall objectives of the course remain the same as its predecessor, but with a deeper and stronger level of exposition.

Course learning outcomes

At the end of the course, students will be able to:

– design of relational database management systems (RDBMs)

– understand and apply know how different index and storage structures are implemented.

– Understand and apply how queries are optimized and processed.

– Understand how multi-user concurrency control works in modern DBMSs.

– Understand and apply logical query languages like Datalog to query databases

Description

1. Data Storage 1.1 Datatypes in SQL 1.1.1 Fixed-Length Datatypes 1.1.2 Variable-Length Datatypes 1.2 Records 1.2.1 Fixed-Length Records 1.2.2 Variable-Length Records 1.3 Blocks & Addressing 1.4 Data Modifications 1.5 Five-Minute Rule 2. Indexing 2.1 Primary Indexes vs. Secondary Indexes 2.2 B+ Trees 2.3 Hashing-based Indexes 2.3.1 Extensible Hashing 2.3.2 Linear Hashing 2.4 Multi-Dimensional Indexes 2.4.1 KD Trees 2.4.2 Quad Trees 3. Query Optimization 3.1 Physical Query Operators 3.1.1 External Sorting 3.1.2 Joins (Nested Loops, Merging, Hashing) 3.2 Cost Models for Query Optimization 3.3 Join-Order Optimization & Dynamic Programming 4. Concurrency Control 4.1 Transactions 4.1.1 Acid Properties of Transactions 4.1.2 Schedules & Serializability 4.2 Transaction Manager 4.2.1 Types of Conflicts 4.2.2 Conflict Resolution 4.2.3 Conflict-Serializability 4.3 2-Phase Locking 4.4 Optimistic Concurrence Control 5. Datalog 5.1 Non-Recursive Datalog: Syntax & Interpretation 5.1.2 Evaluation Strategies 5.1.3 Safe Rules 5.1.4 Relationship to Relational Algebra 5.2 Recursive Datalog 5.2.1 Evaluation Strategies (Naïve vs. Semi-Naive) 5.2.2 Recursion in SQL 5.2.3 Negation & Stratifiability 6. Data Warehousing (OLAP vs. OLTP) 6.1. OLAP Schemas (Fact vs. Dimension Tables) 6.2 Data Cubes 6.3 Business Intelligence 6.3.1 Frequent Itemset Mining 6.3.2 A-Priori Algorithm in SQL

Assessment

Assessment modality:

Combined assessment

Assessment tasks

Task 1: Written or oral midterm exam (50%)Grading scheme: 20 points (0-20)Objectives: Reproduction and application of the course contentsAssessment rules: Open book written exam (whole class) or 30 minutes individual oral exam.Assessment criteria: Exam results are based

on the correctness of the provided answers. Task 2: Written or oral final exam (50%)Grading scheme: 20 points (0-20)Objectives: Reproduction and application of the course contentsAssessment rules: Open book written exam (whole class) or 30 minutes individual oral exam.Assessment criteria: Exam results are based on the correctness of the provided answers.Task 3: Active participation – Presentation of exercise solutions – Up to 2 bonus pointsGrading scheme: 20 points (0-20)Objectives: Solution of exercise problemsAssessment rules: A correct presentation (in class) of the solution to one problem of an exercise sheet yields one bonus point. Up to two bonus points can be achieved per student this way per semester. Bonus points are added on top of the points the student has achieved through the exams. At most 20 points can be achieved in altogether.Assessment criteria: Correctness of the provided solutions.Task 4: Written RETAKE exam (100%)Grading scheme: 20 points (0-20)Objectives: Reproduction and application of the course contentsAssessment rules: Closed book written exam (whole class) or 30 minutes individual oral exam.Assessment criteria: Solution of exercise problems
Note
Syllabus

☒Yes ☐No

Remarks:

Available on Moodle.

Literature list

☒Yes ☐No

Remarks:

Database Systems: The Complete Book (ISBN: 978-0131873254)

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=3576

Other, please specify:

PROGRAMMING FUNDAMENTALS 4

Objectives

Programming languages are a key instrument in computer science: they are used to write programs executed by a computer either through compilation or interpretation. They provide programmers with different abstraction mechanisms that enable rapid (and reliable) software development, while hiding the complexity of the underlying hardware. In general, they can be seen as communication tools not only to communicate with computers but also with humans. The objective of this course is to help students understand the concepts underlying the main programming languages, as well as their capabilities and limitations.

Course learning outcomes

At the end of the course, students will be able to:•Understand the most common programming language paradigms.•Explain the main concepts of programming languages.•Implement the main features of a programming language (e.g., type inference)•Analyse the features of a new programming language.•Select the most suitable programming language for a new project.

Description

Course content 1. Introduction, history of programming languages 2. Programming Language Concepts 3. Lambda calculus 4. Binding, Parameter passing modes, 5. The Algol family of languages 6. ML 7. Types systems and type inference 8. Scope, Functions, and Storage Management 9. Control structures and exceptions 10. Modularity and abstraction 11. Object Orientation 12. Concurrent Programming 13. Summary and Review

Assessment

Assessment modality: Combined assessmentAssessment tasksTask 1: Written exam (60%)Grading scheme: 20 points (0-20)Objectives: Assessing the students' understanding of the main concepts presented in the lectures and their capability to solve some exercises based on the algorithms presented in class.Assessment rules: The use of documents (including books and notes) is not authorized. The use of electronic devices is not authorizedAssessment criteria: Correctness of the answersTask 2: Take-home assignment (20%)Grading scheme: 20 points (0-20)Objectives: Programming exercises to show the students can implement fundamental algorithms of programming language designAssessment rules: Use of generative AI technologies must be disclosed upon submission; the prompts used to get the results should be included in the submission. Late submissions are penalized by 50% (<24h), 75% (>=24h and < 48h), 100% (>=48h)Assessment criteria: Correctness of the program functionality, overall program design, coding style.Task 3: Presentation (20%)Grading scheme: 20 points (0-20)Objectives: By the end of the course, the student should be able to analyse the features of a previously unknown programming language.Assessment rules: Each student will be asked to present the main concepts of a programming language chosen from a list proposed by the instructor.Assessment criteria: Depth and breadth of the presentation, quality of the visual support (slides)Task 4: Retake (60%)Grading scheme: 20 points (0-20)Objectives: A student may retake the written exam. No resitting is foreseen for the take-home assignments and the final presentation.Assessment rules: Same as task 1 Assessment criteria: Same as task 1

Note

Syllabus

☒Yes ☐No

Remarks:

Available on Moodle

Literature list

☒Yes ☐No

Remarks:

Concepts in programming languages /John C. Mitchell. 2003. Cambridge University Press.Available on A-Z.lu

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=6843

THEORETICAL COMPUTER SCIENCE 2

Objectives

The course aims to provide the students an introduction to formal methods and basic concepts, theories, logics, and verification algorithms for automata-based model checking.

Course learning outcomes

After the course, the students will be able to:-model computer systems as transition systems,-understand what types of properties can be specified on the transition system, -formalise such properties in logics (LTL and CTL), -verify whether a transition system satisfies a property written as a logic formula.

Description

Modelling computer systems as transition systems (hardware circuits, sequential programs, concurrency and communication, shared variables, message-passing, channel systems).Linear-time properties (invariants, safety, liveness, and fairness).Regular safety properties, Buchi automata, omega-regular properties.Linear temporal logics (syntax, semantics, equivalence).LTL model checking algorithms.Computation tree logic (syntax, semantics, equivalence).CTL model checking algorithms.

Assessment

Assessment modality:

Combined assessment

Assessment tasks

Task 1: Written exam (40%)Grading scheme: 20 points (0-20)Objectives: The students have fulfilled the objectives of the course.Assessment rules: The students should perform the exam independently by themselves.Assessment criteria: Quality of the submitted exam.Task 2: Take-home assignment (40%)Grading scheme: 20 points (0-20)Objectives: The students can apply the course materials to finish the assignments/homework.Assessment rules: The students should perform the assignments independently by themselves. Assessment criteria: Quality of the submitted assignments/homework.Task 3: Attendance (20%)Grading scheme: 20 points (0-20)Objectives: The students need to be present at the classes in order to understand the basic concepts, theories and verification algorithms.Assessment rules: Be present at both lectures and practical sessions. Otherwise, inform the teachers about the reasons of absence. Assessment criteria: Be present and active at both lectures and practical sessions.Task 4: Written RETAKE exam (100%)Grading scheme: 20 points (0-20)Objectives: The students have fulfilled the objectives of the course.Assessment rules: The students should perform the exam independently by themselves.Assessment criteria: Quality of the submitted exam.

Note

Syllabus

☐Yes ☒No

Remarks:

Literature list

☒Yes ☐No

Remarks:

Baier, C., Katoen, J. (2008). Principles of Model Checking. United Kingdom: MIT Press.

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=3577

Other, please specify:

PROGRAMMING LANGUAGES

Objectives

This course aims to make you understand programming language implementation and formal language processing in an as easy as possible way through concrete examples. It will guide you through all the main phases of the design and the implementation of an interpreter and/or traditional compiler

Course learning outcomes

The course aims to enable you to design and implement compilers and similar language processing tools. To be able to design and implement interpreters and compilers will:•make you a better programmer as you will better understand a language's intricacies. •make you a better computer scientist, because programming technologies span so many areas of the discipline, including formal language theory, grammars, computability, and semantics all the advanced concepts in modern programming languages.•allow you to practice principles and tools seen in previous semesters, as interpreters and compilers are generally large and complex software.•let you get very quickly into the business of implementing a programming language and running programs written in it.

Description

Code Generation Techniques/Schemes for Imperative as well as Object-Oriented Languages, Lexing, Lexer Generation, Parsing, Parser Generation, Static (Code) Analysis, Compiler Optimizations, SCADE/Lustre.

Assessment

Assessment modality:

Continuous assessment

Assessment tasks:

Task 1: Take-home assignment (20%)Grading scheme: 20 points (0-20)Objectives: Code GenerationAssessment rules: All auxiliary materials allowed, but no group work. Assessment criteria: Code generation follows strict translation schemes. This task requires you to apply these schemes to a C program to yield a semantically equivalent assembly program. Submitted homework (code) is graded based on whether the required translation schemes have been correctly applied. Task 2: Take-home assignment (20%)Grading scheme: 20 points (0-20)Objectives: Lexing and Lexer Generation: Formally specify tokens and token classes and construct a lexer for these tokens / token classesAssessment rules: All auxiliary materials allowed, but no group work. Assessment criteria: Lexer generation follows a sequence of well-defined transformations/algorithms. Grading is based on whether these algorithms have been correctly applied and whether the tokens themselves have been correctly specified using regular expressions.Task 3: Take-home assignment (20%)Grading scheme: 20 points (0-20)Objectives: Top-Down Parsing and Parser Generation: Construct a recursive descent parser and an LL parser for given context-free grammarsAssessment rules: All auxiliary materials allowed, but no group work. Assessment criteria: Top-down parser generation yields an action table (LL) or method implementations (RD). The grade is determined by the

correctness of the action table and correctness and coding style of the methods, respectively.Task 4: Take-home assignment (20%)Grading scheme: 20 points (0-20)Objectives: Bottom-Up Parsing and Parser Generation: Construct an LR parser for a given context-free grammarAssessment rules: All auxiliary materials allowed, but no group work. Assessment criteria: Bottom-up parser are given by an action and goto tables (LR), as an intermediate step a so-called Item PDA may be generated. The grade is determined by the correctness/completeness of this automaton and the correct derivation of the action/goto table from the automaton.Task 5: Take-home assignment (20%)Grading scheme: 20 points (0-20)Objectives: Static Program Analysis: Design/definition of a sound static analysisAssessment rules: All auxiliary materials allowed, but no group work. Assessment criteria: Sound static program analyses are defined by a complete lattice (of potential analysis results), edge effects for the various code constructs, a MOP definition, etc. The grade is determined by the correctness (soundness and applicability) of your definitions for an appropriate lattice/MOP, edge effects etc.
Note
Syllabus

☐Yes ☒No

Remarks:

Literature list

☒Yes ☐No

Remarks:

Lecture slides contain up-to-date literature list.

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=3578

Other, please specify:

INTELLIGENT SYSTEMS 1

Objectives

This course aims to offer a foundation, ideas, and techniques underlying the design of intelligent agents and their application in various real-world domains. Also, it offers different ways of system implementations with “intelligent” functionality through statistical and decision-theoretic modelling paradigms where agents will draw inferences in uncertain environments and optimize actions for arbitrary reward structures. Students will learn to recognize when intelligent functionality and artificial intelligence may be a good solution to a problem and to select appropriate AI methodologies and strategies. Further, they will acquire knowledge enabling them to develop the necessary skills to design and implement an intelligent system.

Course learning outcomes

Learning outcomesAt the end of the course, students will be able to:•Describe the concept of single/multi-agent systems and their implementations.•Model various search algorithms with different problem types.•Determine the most adequate search solution/algorithm in each problem scenario.•Design autonomous agents that efficiently make decisions in fully informed, partially observable, and adversarial settings.

Description

The course covers the following topics:1. Fundamental Issues1.1. Overview of AI problems, examples of successful recent AI applications.1.2. What is intelligent behavior? The Turing test, Rational versus non-rational reasoning.1.3. Problem characteristics: Fully versus partially observable, Single versus multi-agent, Deterministic versus stochastic, Static versus dynamic, Discrete versus continuous.1.4. Nature of agents: Autonomous versus semi-autonomous, Reflexive, goal-based, and utility-based, the importance of perception and environmental interactions.1.5. Philosophical and ethical issues. 2. Basic Search Strategies2.1. Problem spaces (states, goals and operators), problem solving by search.2.2. Factored representation (factoring state into variables).2.3. Uninformed search (breadth-first, depth-first, depth-first with iterative deepening).2.4. Heuristics and informed search (hill-climbing, generic best-first, A*).2.5. Space and time efficiency of search.2.6. Two-player games (introduction to minimax search).2.7. Constraint satisfaction (backtracking and local search methods.3. Basic Machine Learning3.1. Definition and examples of broad variety of machine learning tasks, including classification.3.2. Inductive learning.3.3. Simple statistical-based learning, such as Naive Bayesian Classifier, decision trees.3.4. The over-fitting problem.3.5. Measuring classifier accuracy.

Assessment

Assessment modality:

Combined assessment

Assessment tasks

Task 1: Written exam (40%)Grading scheme: 20 points (0-20)Objectives: To assess the student’s comprehension of the taught topics during the lecture part of the course. Assessment rules: Devices of any kind are not allowed; it is a closed-book exam. Assessment criteria: The

exam contains several questions with a preset marking scheme and marks distribution for each question. Task 2: Take-home assignment (40%)Grading scheme: 20 points (0-20)Objectives: To assess students' psychomotor skills in implementing and designing intelligent agents throughout the training sessions. Assessment rules: All resources are allowed Assessment criteria: Project work / Assignment reports Task 3: Active participation (20%)Grading scheme: 20 points (0-20)Objectives: To encourage the students to be involved in the discussions and critical thinking during both lecture and training sessions. Assessment rules: N/A Assessment criteria: Attendance and question answering throughout lecture and training sessions. Task 4: Written RETAKE exam (100%)Grading scheme: 20 points (0-20) Objectives: Retake the course in case of failure or missing the written exam Assessment rules: Devices of any kind are not allowed; it is a closed-book exam. Assessment criteria: The exam contains several questions with a preset marking scheme and marks distribution for each question.
Note
Syllabus

☒Yes ☐No

Remarks:

Available on Moodle.

Literature list

☒Yes ☐No

Remarks:

Artificial Intelligence: A Modern Approach (Prentice Hall Series in Artificial Intelligence) by Stuart Russell and Peter Norvig http://aima.cs.berkeley.edu/

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=3579

BACHELOR SEMESTER PROJECT 4

Objectives

During a standard BSP, students discover research and development domains, produce concrete artefacts related to computer science knowledge areas covered in the BICS, collaborate with their academic advisor(s), learn new technologies related to computer science, learn new knowledge related to computer science, apply the scientific and technical knowledge learned during the BICS, apply the primary and secondary languages knowledge learned during the BICS. The course content and learning objectives are consistent with the BiCS programme content that the student is following.

Course learning outcomes

At the end of the BSP, the student should have acquired specific learning outcomes defined on an individual basis between the academic advisor(s) and him/herself. The learning outcomes should be agreed upon at the beginning of each project and be aligned with the general BiCS curriculum

Description

Specific course content is defined on an individual basis between the academic advisor(s) and the student for each project.

Assessment

Assessment modality: Combined assessment

Assessment tasks

Task 1: Final report and in-person presentation (100%)Grading scheme: 20 points (0-20) Objectives: Assess the quality and quantity of work done during the semester. Assessment rules: Time constrained submissions. Assessment criteria: The detailed criteria are specific to each project subject and applied on the following weighted deliverables: technical & scientific quality of the final report, primary language (60%) technical & scientific quality of the final in-person presentation, primary language (30%) technical & scientific quality of the final report, secondary language summary (5%) technical & scientific quality of the final in-person presentation, secondary language summary (5%) Task 2: RETAKE (100%)Grading scheme: 20 points (0-20Objectives: Option to repeat a previously failed project (with a final assessment of <10 points) with the same topic and under the same supervisor. Assessment rules: The same deliverables as for the previously failed project must be resubmitted under the given time constraints. Assessment criteria: Same as for Task 1.

Note

Syllabus

☒Yes ☐No

Remarks:

Accessible from the Moodle course page

Literature list

☒Yes ☐No

Remarks:

Accessible from the Moodle course page

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=3581

5th Semester
COMPUTATIONAL SCIENCE
Objectives
Numerous problems in engineering, physical and economical industries and application domains essentially boil down to minimising a single function: the objective function in optimization terminology. Minimisation is not only the basis of many simulation tools, but also the basis of many parameter identification approaches. Unfortunately, there is not one minimisation method that outperforms the others. In this module, the student will therefore become familiar with three numerical minimisation techniques, each with its own advantages and disadvantages. Furthermore, three ways to deal with constraints in minimisation problems will be considered, also each with its own advantages and disadvantages. The student will implement the minimisation techniques herself/himself in a programming language of her/his choice, hereby effectively implementing her/his own simulations. In this way, the student will truly be exposed to the methods' advantages and disadvantages, and she/he will capture the relevant technical complexities of the methods. Thus, the aim of the module is to teach the student a variety of unconstrained and constrained minimisation approaches and understand their beneficial and disadvantageous differences.

Course learning outcomes
At the end of the course, the student will be able to:•Understand and be able to work with descent methods. •Understand and be able to work with Newton's method in optimization. •Understand and be able to work with quasi-Newton methods in optimization. •Understand and be able to incorporate constraints in objective functions using substitution. •Understand and be able to incorporate constraints in objective functions using the penalty method. •Understand and be able to incorporate constraints using the method of Lagrange multipliers.
Description
1. Descent methods: steepest descent method, line search using the Armijo rule, conjugate gradient method, compute multivariate derivatives, implement the methods.2. Newton's method: Compute multivariate second-order derivatives, solve linear systems, implement the method.3. Quasi-Newton methods: line search using the Wolfe conditions, implement the BFGS method and the L-BFGS method.4. Implement constraints in previous unconstrained objective functions using substitution.5. Implement constraints in previous unconstrained objective functions using the penalty method.6. Implement constraints in previous unconstrained objective functions using the method of Lagrange multipliers.
Assessment
Assessment modality: Combined assessmentAssessment tasksTask 1: Written exam (20%)Grading scheme: 20 points (0-20)Objectives: Assess the student's understanding of (1) descent methods and (2) Newton's method.Assessment rules: The lecture notes and even the internet may be used. However, any means of communication is forbidden.Assessment criteria: The student must use its own implementations, made during the semester, to calculate some minimization problems. Open questions may also be posed, which require a textual response.Task 2: Written exam (20%)Grading scheme: 20 points (0-20)Objectives: Assess the student's understanding of (3) quasi-Newton methods.Assessment rules: The lecture notes and even the internet may be used. However, any means of communication is

forbidden.Assessment criteria: The student must use its own implementations, made during the semester, to calculate some minimization problems. Open questions may also be posed, which require a textual response.Task 3: Written exam (60%)Grading scheme: 20 points (0-20)Objectives: Assess the student's understanding of (1) descent methods, (2) Newton's method, (3) quasi-Newton methods, (4) constraint incorporation using substitution, (5) constraint incorporation using the penalty method, (6) constraint incorporation using the method of Lagrange multipliers.Assessment rules: The lecture notes and even the internet may be used. However, any means of communication is forbidden.Assessment criteria: The student must use its own implementations, made during the semester, to calculate some minimization problems. Open questions may also be posed, which require a textual response.Task 4: Written exam – RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: Assess the student's understanding of (1) descent methods, (2) Newton's method, (3) quasi-Newton methods, (4) constraint incorporation using substitution, (5) constraint incorporation using the penalty method, (6) constraint incorporation using the method of Lagrange multipliers.Assessment rules: The lecture notes and even the internet may be used. However, any means of communication is forbidden. In principle, again the resit exams consist of 2 midterm exams and one final exam – with the same weights for the final grade as for the standard exams.Assessment criteria: The student must use its own implementations, made during the semester, to calculate some minimization problems. Open questions may also be posed, which require a textual response.
Note
Course materials

Syllabus☒Yes☐NoRemarks:Available on the Moodle page.Literature list☒Yes☐NoRemarks:Lecture notes are provided by the instructor.Moodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=4194

SOFTWARE ENGINEERING 1

Objectives

Software engineering is the discipline concerned with the application of theory, knowledge, and practice to effectively and efficiently build reliable software systems that satisfy the requirements of customers and users. This discipline is applicable to small, medium, and large-scale systems. It encompasses all phases of the lifecycle of a software system, including requirements elicitation, analysis and specification; design; construction; verification and validation; deployment; and operation and maintenance." [ACM/IEEE-CS Joint Task Force on Computing Curricula. 2013. Computer Science Curricula 2013. ACM Press and IEEE Computer Society Press. DOI: http://dx.doi.org/10.1145/2534860].

The objectives of this course are to provide knowledge on the discipline such that the student understands the main aspects of it. The course focuses on software processes, project management, quality standard and requirements engineering for a deep understanding of this development phase.

Course learning outcomes

At the end of the course, the student will be able to:•LO1/(MF) Apply consistent documentation and program style standards that contribute to the readability and maintainability of software. •LO2/(M) Describe the relative advantages and disadvantages among several major process models (e.g., waterfall, iterative, and agile).•LO3/(M) Describe the different practices that are key components of various process models.•LO4/(M) Differentiate among the phases of software development.•LO5/(M) Explain the concept of a software lifecycle and provide an example, illustrating its phases including the deliverables that are produced.•LO6/(M) Compare several common process models with respect to their value for development of classes of software systems taking into account issues such as requirement stability, size, and non-functional characteristics.•LO7/(M) Define software quality and describe the role of quality assurance activities in the software process.•LO8/(M) Describe the intent and fundamental similarities among process improvement approaches.•LO9/(M) Use project metrics to describe the current state of a project.•LO10/(M) Using a particular software process, describe the aspects of a project that need to be planned and monitored, (e.g., estimates of size and effort, a schedule, resource allocation, configuration control, change management, and project risk identification and management). •LO11/(MF) Track the progress of some stage in a project using appropriate project metrics. •LO12/(MF) List the key components of a use case or similar description of some behaviour that is required for a system.•LO13/(M) Describe how the requirements engineering process supports the elicitation and validation of behavioural requirements. •LO14/(MF) Interpret a given requirements model for a simple software system.•LO15/(M) Describe the fundamental challenges of and common techniques used for requirements elicitation. •LO16/(MF) List the key components of a data model (e.g., class diagrams or ER diagrams).•LO17/(MF) Identify both functional and non-functional requirements in a given requirements specification for a software system.•LO18/(MF) Conduct a review of a set of software requirements to determine the quality of the requirements with respect to the characteristics of good requirements. •LO19/(MF) Apply key elements and common methods for elicitation and analysis to produce a set of software requirements for a medium-sized software

system.•LO20/(MF) Use a common, non-formal method to model and specify the requirements for a medium-size software system.•LO21/(MF) Translate into natural language a software requirements specification (e.g., a software component contract) written in a formal specification language. •LO22/(MF) Create a prototype of a software system to mitigate risk in requirements. •LO23/(MF) Understand and apply the Messir requirement engineering to a small size application development process.•LO24/(M) Understand and apply the Messir use-case model in the context of the development methodology.•LO25/(M) Understand and apply the Messir environment model in the context of the development methodology.•LO26/(MF) Understand and apply the Messir concept model in the context of the development methodology.•LO29/(F) Understand and apply the UML use-case diagram.•LO30/(F) Understand and apply the UML sequence diagram.•LO31/(MF) Understand and apply the UML class diagram.•LO32/(MF) Understand and apply the Messir operation model at definition level in the context of the development methodology.•LO33/(MF) Understand and apply the Messir operation model at specification level in the context of the development methodology.Remark: the letter M indicates that the learning outcome will be assessed at midterm exam, F at final exam assessment task, and MF at both).

Description

• Program correctness: types of errors (syntax, logic, run-time), the concept of a specification, defensive programming (e.g., secure coding, exception handling), code reviews, testing fundamentals and test-case generation, the role and the use of contracts (including pre- and post-conditions), unit testing.• Simple refactoring. • Modern programming environments: Code search, Programming using library components and their APIs.• Documentation and program style. • Introduction to software process models (e.g., waterfall, incremental, agile); activities within software lifecycles. • Programming in the large vs. individual programming. • Evaluation of software process models. • Software quality concepts. • Process improvement. • Software process capability maturity models. • Software process measurements.• Software quality assurance and the role of measurements. • Release management. • Requirements analysis and design modelling tools.• Describing functional requirements using, for example, use cases or users' stories. • Properties of requirements including consistency, validity, completeness, and feasibility. • Software requirements elicitation.• Describing system data using, for example, class diagrams or entity-relationship diagrams. • Non-functional requirements and their relationship to software quality (cross-reference IAS/Secure Software Engineering).• Evaluation and use of requirements specifications. • Requirements analysis modelling techniques.• Acceptability of certainty/uncertainty considerations regarding software/system behaviour.• Prototyping. • Basic concepts of formal requirements specification.• Requirements specification.• Requirements validation. • Requirements tracing. • System design principles: levels of abstraction (architectural design and detailed design).• Structural and behavioural models of software designs.• Relationships between requirements and designs: transformation of models, design of contracts, invariants.

Assessment

Assessment modality: Combined assessmentAssessment tasks

Task 1: Written exam – MIDTERM (40%)Grading scheme: 20 points (0-20)Objectives: Assess to which extend the targeted course learning outcomes have been acquiredAssessment rules: no

document allowed – no electronic device allowedthe duration of the exam can be up to FOUR successive hoursAssessment criteria: A set of questions are asked to the student on a subset of the learning outcomes targeted by the assessment task (cf. learning outcomes). Task 2: Written exam – FINAL (60%)Grading scheme: 20 points (0-20)Objectives: Assess to which extend the course learning outcomes have been acquiredAssessment rules: no document allowed – no electronic device allowed. The duration of the exam can be up to FOUR successive hoursAssessment criteria: A set of questions are asked to the student on a subset of the targeted learning outcomes by the assessment task (cf. learning outcomes).Task 3: Written exam – RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: Assess to which extend the course learning outcomes have been acquired for a student having ALREADY FAILED ONCE TO THE COURSE USING THE assessment tasks 1 AND 2.Assessment rules: no document allowed – no electronic device allowed. The duration of the exam can be up to FOUR successive hoursAssessment criteria: A set of written questions are asked to the student in order to evaluate the acquisition level reached for a subset of the learning outcomes.The final grade FOR THE Task3 – RETAKE assessment is computed as follows:- 40% the > MIDTERM grade- 60% the grade of the RETAKE written questions answersThe MIDTERM assessment task 1 can be re-executed by a redoing students only if the student registers to the course during the WINTER semester.
Note
Course materials

Syllabus☒Yes☐NoRemarks:Accessible from the Moodle course pageLiterature list☒Yes☐NoRemarks:Accessible from the Moodle course pageMoodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=3916

INTRODUCTION TO IOT

Objectives

Students will be equipped with competences for the development of applications for the Internet of Things (IOT):Know the challenges of IoT networks considering the various specificities and application constraints (e.g. QoS, scalability, real-time application, operational safety, etc.).Master the basics of IoT communication protocols (Application layer).Master the selection and implementation of different IoT communication protocols.Design, analyze and criticize different data collection, storage and processing data architectures, both their possibilities and their limits.Design one or more IoT applications using machine learning modules based on the collected data.

Course learning outcomes

On successful completion of this course, students are capable to:Define and develop the functional and technical specifications of a network and telecom equipment (hardware, software, implementation, etc.).Determine architecture components, technologies, equipment, tools supports and integrate them according to the specifications.

Description

Introduction to issues relating to the interoperability of IoT networks/protocols (application layer – OSI)Real-time networks: Profibus, Modbus, Modbus-TCPIOT protocols on application layer: HTTP (REST API), MQTT, CoAP, OneM2M, O-MI/O-DFNOSQL databases: MongoDB, ElasticSearchGetting started with Node-Red (visual programming tool – open source – developed by IBM) for IoT application development:data collection: Arduino, & sensors, Cloud API endpoints…data storage: databases (SQL, NoSQL)data treatment: Node-Red (JavaScript)publication of data via dashboard (H2M) and machine interfaces (M2M): Implementation of an HTTP server (REST API specification, server deployment)

Assessment

First time students:40%: practical group project60%: final written examRepeating students:100% final written exam

Note

Literature: Relevant literature will be provided during the lecture.

HUMAN-COMPUTER INTERACTION (HCI)

Objectives

The course provides students with a comprehensive understanding of the fundamental principles and concepts underlying Human-Computer Interaction, such as research methods, user interfaces, or interaction techniques.

Course learning outcomes

1.Understand fundamental HCI principles and their role in designing interactive systems.2.Compare and contrast different user-centred design and research methodologies.

Recognize the societal implications of HCI, including e.g. privacy, security, and accessibility.

Description

1. Introduction1.1 Why is HCI challenging?1.2 Human-computer interaction as a field1.3 Fundamental concepts1.4 Why HCI matters1.5 Our approach to HCI2. Understanding people2.1 Types of understanding2.2 Areas of understanding2.3 Applying our understanding of people2.4 Is a general understanding of people possible?3. Perception3.1 Sensory modalities3.2 Elementary functions of perception in HCI3.3 Visual perception and attention3.4 Perception is an active process4. Motor Control4.1 Elements of a motor control task in HCI4.2 Target Acquisition4.3 Simple Reactions4.4 Choice reaction4.5 Gesturing5. Cognition5.1 General Findings about Cognition5.2 Cognitive control5.3 Memory and Learning5.4 Reasoning and Decision-Making5.5 Simulating cognition in interactive tasks6. Needs and Motivations6.1 Psychological Needs6.2 Motivations6.3 Behavior Change6.4 Gamification7. Experience7.1 What is Experience?7.2 Types of Experience7.3 Assessing and Measuring Experience7.4 Can experiences be designed?8. Collaboration8.1 Understanding collaboration8.2 Coordination8.3 Group cognition8.4 Why collaborative systems fail9. Communication9.1 Beyond face-to-face communication9.2 Conversations9.3 Online behavior9.4 Social Networks and Online Communities9.5 Computers as Communication Partners10. User Interfaces10.1 Definition and Elements10.2 Interaction Styles10.3 Design Objectives10.4 Design Space Analysis11. Input Devices11.1 Principles of Sensing11.2 Keypads and Keyboards11.3 Pointing Devices11.4 Uncertain Control11.5 Expanding the Limits of Sensing12. Displays12.1 Encoding and Rendering12.2 Simple Displays12.3 Visual Displays13. Contents13.4 Audio13.5 Haptics13.6 Emerging Forms of Displays14. Interaction Techniques14.1 Objectives for Interaction Techniques14.2 Pointing techniques14.3 Selection and manipulation techniques14.4 Text entry14.5 Camera control14.6 Two perspectives: Control and Learnability15. Commands and Navigation15.1 Naming Objects and Actions15.2 Command-Line Interfaces15.3 Organizing Information15.4 Menu User Interfaces15.5 Notational Systems15.6 Recognition versus Recall16. Graphical User Interfaces16.1 A Brief History of the GUI16.2 Design Objectives16.3 The Principle of Direct Manipulation16.4 Anatomy of a Graphical User Interface16.5 Designing a Graphical User Interface16.6 Why do we still have GUIs?17. Reality-Based Interaction17.1 Mobile User Interfaces17.2 Ubiquitous Computing17.3 Tangible User Interfaces17.4 Mixed Reality17.5 Should we imitate reality or go beyond it?

Assessment

Assessment modality: Combined assessmentAssessment tasksTask 1: Assignments (40%)Grading scheme: 20 points (0-20)Objectives: Students will get familiarized with HCI

research through readings of academic papers.Assessment rules: There will be a list of research papers available in Moodle. Every week, students must present one of those papers in front of the class. There will be a monthly or bi-monthly rotation, depending on the number of students enrolled in the course, which means that each student will present two or three papers in total.Assessment criteria: Each presentation will be graded using a 0–10-point scale and the final grade in this part will be the average grade of all presentations delivered by the student. Students must get an average grade of at least 5 points (out of 10) in this part in order to qualify for the final exam. If a student scores less that 5 points in this part, the final grade in the course will be the minimum grade achieved in either the assignments or the final exam. Any form of cheating or plagiarism will be penalized with a grade of 0 in this part, which will make the student to fail the whole course. The grade in this part will be retained until the student passes the final exam, unless the student decides to repeat it in the next academic year. There is no retaking option for this part within the same academic year, which means that a student who fails this part must enroll again in the course in the next academic year. Retaking students must inform the course responsible whether they want to keep their grade from the previous year.Task 2: Final exam (60%)Grading scheme: 20 points (0-20)Objectives: Students will demonstrate that they have acquired the required competencies according to the theory covered in the course.Assessment rules: Only students who got at least 5 points in the assignments should take the final exam, otherwise they are advised to de-enrol from the final exam in order to avoid missing an official attempt. No supporting materials are allowed in the exam, i.e. no cheatsheet, no books, no mobile devices, no laptops, etc.Assessment criteria: The exam is a paper-based quiz with 20 questions that must be solved in 30 minutes. Each question comes with 4 possible answers, among which only one is the right one. A student must get at least 5 points (out of 10) in the final exam in order to pass the course. If a student scores less that 5 points in this part, the final grade in the course will be the minimum grade achieved in either the assignments or the final exam. Any form of cheating or plagiarism will be penalized with a grade of 0 in this part, which will make the student to fail the whole course.The grade in this part will not be retained for the next year. Failing students can retake the exam in the next examination period or enroll again in the course the next academic year.
Note
Course materials

Syllabus☒Yes☐NoRemarks:Available on MoodleLiterature list☒Yes☐NoRemarks:Reference texbook:•Kasper Hornbæk, Per Ola Kristensson, Antti Oulasvirta. Introduction to Human-Computer Interaction. Oxford University Press. 2024.Moodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=3917 Other, please specify:Class attendance is expected for the practical sessions, otherwise the students will fail the continuous evaluation.

INTRODUCTION TO MACHINE LEARNING

Objectives

This course introduces Machine Learning (ML) principles and its three main learning paradigms (supervised, unsupervised, and reinforcement learning). For each learning paradigm, it presents some of its most typical foundational models and discuss them from the perspective of representation, evaluation, and optimization. A special attention is given to a basic introduction into deep learning techniques and generalization. The course mixes theoretical concepts with vanilla implementations of various ML models.

Course learning outcomes

At the end of the course the student will be able to:• define Machine Learning and differentiate the terms AI, Machine Learning, and Deep Learning.• describe the differences among the three Machine Learning paradigms: supervised, unsupervised, and reinforcement learning.• determine which of the three Machine Learning paradigms is appropriate to a particular type of problems.• derive, implement, and evaluate some of the most typical Machine Learning models. • derive, implement, and evaluate some basic Deep Learning models and their learning algorithms.• explain proper ML evaluation procedures, including the differences between training and testing performance.• apply Machine Learning models to real-world problems.• identify overfitting in the context of a problem and describe solutions to overfitting.• evaluate the performance of a Machine Learning algorithm on a real-world dataset.

Description

The following topics are covered in the course:• Basics – ML Introduction • Basics – Data preparation for ML • Supervised Learning – Regression • Supervised Learning – Classification• Unsupervised Learning – Dimensionality reduction• Unsupervised Learning – Clustering• Reinforcement Learning – Preliminaries• Reinforcement Learning – Basic methods• Deep Learning – Learning Deep Representations• Deep Learning – Models• Deep Learning – Deep Reinforcement Learning• Generalization• Research – Glimpse on state-of-the-art research• Engineering – ML and the real-world

Assessment

o Three take-home assignments – 40% of the final grade in total split as follows: 10%, 10%, and 20%o Written exam – 60% of the final grade o Students having failed the course will have to retake the exam at the next exam session. This exam will count for 100% of the final grade.

Note

Literature:• Bishop, C.: Pattern Recognition and Machine Learning, 2006• Goodfellow, I., Bengio, Y. & Courville, A.: Deep Learning, 2016• Sutton, R., Barto, A.: Reinforcement Learning: An Introduction, second edition, 2018• Hastie, T, Tibshirani, R. & Friedman, J.: The Elements of Statistical Learning, 2009• Zhang, A., Smola, A.J., Lipton, Z., Li, M.: Dive into Deep Learning, 2023

NATURAL LANGUAGE PROCESSING

Objectives

This course focuses on the theoretical and practical foundations of Natural Language Processing (NLP), an interesting domain at the intersection of computer science and artificial intelligence. Covering language representation, syntax, semantics, and advanced techniques like deep learning and transformers, the course explores how NLP enables communication with chatbots and other artificial systems. By understanding language models and machine learning's role, students will gain insights into NLP's applications, including sentiment detection, chatbots, and text comprehension. The course aims to equip students with a comprehensive understanding of NLP's significance and motivate them to tackle complex language-related challenges in the field of computer science and AI.

Course learning outcomes

The learning outcomes of this course are summarized as follows: – Comprehension of the theoretical foundations of Natural Language Processing (NLP). – Familiarity with language representation techniques in NLP. – Hands-on experience in solving real-world problems using NLP tools and libraries. – Proficiency in using Python as the primary programming language for NLP tasks. – Familiarity with frameworks like PyTorch and TensorFlow for NLP applications. – Ability to work with Jupyter Notebooks, NLTK, and spaCy for essential "classical" NLP tasks. – Ability to use some machine learning and deep learning algorithms to solve NLP applications. – Application of NLP techniques to more advanced tasks, such as sentiment analysis, text summarization, and question answering systems. Exploration of natural language understanding, including ChatGPT prompting and engineering applications with Langchain.

Description

– Week 1: Introduction to Natural Language Processing – Week 2: Text Preprocessing – Weeks 3-5: Text Classification – Week 6: Vector Semantics – Week 7: Language Modeling – Week 8: Text Representation – Week 9: Introduction to Neural Networks – Weeks 10-12: ChatGPT prompting – Weeks 13- 14: Projects Presentation and Exam preparation

Assessment

Assessment modality: Combined assessmentAssessment tasksTask 1 : Written exam (50)Grading scheme: 20 points (0-20)Objectives: Main test for assessing proficiency of students. Assessment rules: Students will get exercises that test their understanding of the course. Only open books and cheat sheets are permitted; no electronic devices are allowed. Assessment criteria: The answers are graded based on correctness, completeness, and clarity. Task 2: Take-home assignment(20%)Grading scheme: 20 points (0-20)Objectives: To familiarize the students with the material and NLP libraries and applications. Assessment rules: Students will receive Jupyter notebooks with various NLP applications and will need to complete the required source code and execute the final notebook. Assessment criteria: The answers are graded based on correctness, completeness, and clarity. Task 3: Final Project (30%)Grading scheme: 20 points (0-20)Objectives: Choose and implement an NLP application. Assessment rules: Students need to choose the topic of their project, implement it, and present it. Assessment criteria: The grade is based on the relevance and the oral presentation of the project. Task 4: Written exam – RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: Test for assessing proficiency of students and help them pass the NLP course Assessment

rules: Students will get exercises that test their understanding of the course. Only open books and cheat sheets are permitted; no electronic devices are allowed. Assessment criteria: The answers are graded based on correctness, completeness, and clarity.
Note
Course materials

Syllabus☒Yes☐NoRemarks:Available on MoodleLiterature list☒Yes☐NoRemarks:Available on MoodleMoodle page☒Yes☐NoRemarks:https://moodle.uni.lu/course/view.php?id=3921

BACHELOR SEMESTER PROJECT 5

Objectives

During a standard BSP, students discover research and development domains, produce concrete artefacts related to computer science knowledge areas covered in the BICS, collaborate with their academic advisor(s), learn new technologies related to computer science, learn new knowledge related to computer science, apply the scientific and technical knowledge learned during the BICS, apply the primary and secondary languages knowledge learned during the BICS. The course content and learning objectives are consistent with the BiCS programme content that the student is following.

Course learning outcomes

At the end of the BSP, the student should have acquired specific learning outcomes defined on an individual basis between the academic advisor(s) and him/herself. The learning outcomes should be agreed upon at the beginning of each project and be aligned with the general BiCS curriculum.

Description

Specific course content is defined on an individual basis between the academic advisor(s) and the student for each project

Assessment

Assessment modality: Combined assessmentAssessment tasksTask 1: Final report and in-person presentation 100%)Grading scheme: 20 points (0-20)Objectives: Assess the quality and quantity of work done during the semester. Assessment rules: Time constrained submissions. Assessment criteria: The detailed criteria are specific to each project subject and applied on the following weighted deliverables: – technical & scientific quality of the final report, primary language (60%) – technical & scientific quality of the final in-person presentation, primary language (30%) – technical & scientific quality of the final report, secondary language summary (5%) – technical & scientific quality of the final in-person presentation, secondary language summary (5%) Task 2: RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: Option to repeat a previously failed project (with a final assessment of <10 points) with the same topic and under the same supervisor. Assessment rules: The same deliverables as for the previously failed project must be resubmitted under the given time constraints. Assessment criteria: Same as for Task 1.

Note

Course materials

Syllabus ☒Yes☐No Remarks: Accessible from the Moodle course page Literature list ☒Yes☐No Remarks: Accessible from the Moodle course page Moodle page ☒Yes☐No Remarks: https://moodle.uni.lu/course/view.php?id=3334

6th Semester
AI FOR EDUCATION
Objectives
In recent years, the term artificial intelligence (AI) has taken on a new meaning. While the original idea of AI is still to understand and artificially simulate human (cognitive) intelligence, the applications of AI have become increasingly important in recent years. There is currently a real spirit of optimism, more and more new AI companies are being founded, governments, industry and science are investing in research and development projects and in targeted knowledge transfer, and with 'AI for the common good' and 'AI for humans' we are all prepared for future developments. One of these developments relates to education (and training), be it in schools and universities, in industrial training or in the service sector and customer service. The associated opportunities, but also risks, raise the questions of techniques (human-computer interfaces, intelligent systems) of the extent to which AI can be used sensibly and responsibly for learning and knowledge acquisition. AI-based systems should not necessarily be seen as a replacement for existing learning practices and learning techniques, but as a supplement. The spectrum of ideas affected by this is therefore diverse. The aims of this course are to gain an overview of applications in the relevant area and to explore your own ideas.

Course learning outcomes
The aim of the course is for each participant to understand the selected scientific and common research directions in the field of AI/Education (lecture part) and the outline and design of an own meaningful application in this field, which has to be finally presented in a Poster Workshop at the end oif the course.
Description
The theoretical part concerns selected aspects of and takes place in the first part of the semester:• History of AI and selected basics in view of AI/Education• Techniques to be used (Tutoring systems, Expert systems, projects in the AI4Edu focus area, Virtual Reality, etc.• Examples of AI-based teaching and learning methods• Ethical aspects: GDPR, EU AI Act• Future of AI-based Education.The Practical Part concerns a Poster Workshop. It is held in the second part of the semester.The second part concerns a project and its presentation in a poster workshop at the end of the course. All participants work on a project and design a poster reflecting the content of the project. In addition, the project carried out must be summarised in a written document:• Choose of a project topic • Create objectives and work plan• Discuss the data aspect (data sources and software tools; collection and preparation of data for AI models) and discuss the technical environment (e.g. chatbots, recommendation systems, tutoring system, expert systems, etc.)• Implement the project incl. a critical reflection and future work, bibliography • Integration into an educational setting + testing. Analysis of feedback.
Assessment
Assessment modality: Combined assessmentAssessment tasksPart 1 (30h) – AttendanceGrading scheme: Pass/FailObjectives: Click or tap here to enter text. Attendance is compulsory in the lecture part: the candidate qualifies for the poster workshop if he/she has attended at least 8 of the 14 lectures.Pass or FailAssessment rules: The students work independently on the course content. Contact with the teachers is desired.Assessment criteriaPart 2 (30h) – Active participationGrading scheme:20 points (0-20)100%Objectives:

Participants familiarise themselves with the project topic, the terminology and the objectives.Assessment rulesEvaluation •Each group member must attend at least 60% = (8 of 14) lectures and actively contribute to the Poster workshop. If a group member has fulfilled this 60% condition (and only then), the group member qualifies for the Poster Workshop, in which a project idea relating to AI/Education has to be presented.•”Presented” means that 1.Each group contributes to the implementation of this project idea.2.Each group contributes to the design of a poster (at least A3 format) that summarises the most important points of the idea developed. Examples of poster designs can be found, for example, in the MNO building (many posters hang on the walls on the 3rd and 4th floors). The poster must be scientific, not marketing-orientated (more information see below). 3.Presentation of the idea through a talk and demonstrating the implementation in front of a public audience.4.Summarisation of the idea in the form of a written Text (in \LateX) using a template presented (5 pages minimum, English).•The evaluation is as follows:o10% Value of the developed idea. It is important to emphasise the benefits of the idea being developed. Is it realisable and does it make sense? o20% Design of the poster. We follow good practices relating to an effectiveness and expressiveness of the components used (structure of the poster; content of the poster; style like font, colour, etc.)o30% Talk/presentationo40% Written Summary (quality of content, reading and syntax, comprehensibility of the content, completeness of the content, quality of the \LaTeX code, etc.), which will become a chapter in a final booklet.If Generatlve AI tools (ChatGPT and others) are used, then this must be referenced and the prompts listed in a “Table of prompts” next to the Bibliography chapter.
Note
Course materials

Syllabus

☒Yes ☐No

Remarks:

Available on Moodle

Literature list

☒Yes ☐No

Remarks:

Available on Moodle

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=4193

Other, please specify:

SOFTWARE ENGINEERING 2

Objectives

Software engineering is the discipline concerned with the application of theory, knowledge, and practice to effectively and efficiently build reliable software systems that satisfy the requirements of customers and users. The purpose of this course is to address fundamentals and practical areas of software engineering not covered in previous courses. This course will be composed of two parallel tracks, described below. Track 1. Testing and Validation: Testing is the predominant technique used by the software industry to ensure the final software product reliability. It can be roughly compared to a hunt for detecting and eliminating defects and bugs from a software. Thus, testing and validation aim first at assessing the consistency/conformity of a product with respect to its specification, and in practice to ensure at least that it will not crash! These activities are thus crucial and costly for software companies, and eventually aim at providing a controlled level of trust in the final product before delivery to the client (and continuously during maintenance and evolution). Testing is related to all the design stages of the development process and must deal with many application contexts (embedded systems, mobile applications, information systems …) and various levels of complexity (programming-in-the-small, in-the-large and in-the-duration). Besides the fundamentals of software testing, the focus will be on practical techniques that can be applied in real-world software development cycles (agile, continuous integration), including techniques to localize and remove the detected faults. Track 2. Software design and architectures: the design of software systems is largely dependent on the accurate analysis of functional and non-functional requirements. This track will address software architectures, design patterns, and, in general, non-functional requirements. The use cases that we will study will be taken from, or inspired by, real-world industrial software.

Course learning outcomes

At the end of the course the student will:

Remember, and understand the fundamentals of software testing (the concepts and principles behind the many existing techniques).

Apply classical testing techniques, black-box and glass-box ones, on simple programs.

Solve a larger software testing problem (integration and system) and create a basic testing strategy.

Analyze novel enablers (e.g. AI-assistant, APR) and testing challenges (e.g. testing AI-based system).

Understand the main software architectural styles and design patterns

Select the most suitable software architecture and design patterns for a new project.

Description
1. Part 1: Software Testing and Validation 1.1 Fundamentals 1.2. Unit Testing and Diagnosis 1.3. Integration Testing 1.4. Requirements and System Validation 1.5. Transversal aspects to functional testing 2. Track 2: Software Architectures and Non-functional requirements 2.1. Non-functional requirements 2.2. Design patterns 2.3 Software architectural styles
Assessment
Assessment modality: Combined assessment Assessment tasks

Task 1: Written exam – Mid-term exam (40%)Grading scheme: 20 points (0-20)Objectives: Assessing the students' understanding of the main concepts presented in the lectures and their capability to solve some exercises based on the algorithms presented in class. Assessment rules: The use of documents (including books and notes) is not authorized. The use of electronic devices is not authorized. Assessment criteria: Correctness of the answers Task 2: Written exam – Final exam (60%)Grading scheme: 20 points (0-20)Objectives: Assessing the students' understanding of the main concepts presented in the lectures and their capability to solve some exercises based on the algorithms presented in class. Assessment rules: The use of documents (including books and notes) is not authorized. The use of electronic devices is not authorized Assessment criteria: Correctness of the answers Task 3: RETAKE (100%)Grading scheme: 20 points (0-20)Objectives: Assessing the students' understanding of the main concepts presented in the lectures and their capability to solve some exercises based on the algorithms presented in class. Assessment rules: The use of documents (including books and notes) is not authorized. The use of electronic devices is not authorized Assessment criteria: Correctness of the answers
Note
Syllabus

☒Yes ☐No

Remarks:

Available on Moodle

Literature list

☒Yes ☐No

Remarks:

"Introduction to Software Testing" – Paul Ammann and Jeff Offutt – ISBN-13: 9780521880381 – Cambridge Press – 2008"Foundations of Software Testing" – Aditya Mathur – Addison-Wesley Professional – 2007"Software Architecture Patterns" – Mark Richards-O'Reilly, USA-2015

“Design Patterns” Elements of Reusable Object-Oriented Software” – Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides – Addison-Wesley – 1994

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=4191

SECURITY 2

Objectives

The objectives of this course on Information Security are:1) To provide students with comprehensive understanding of public-key cryptography. This will involve studying RSA encryption, signatures, Diffie-Hellman key exchange, along with its secure models, the methods to encrypt and sign securely using RSA, and a study on Public-key Infrastructure, including certificates and SSL protocol.2) To delve into blockchain protocols with a focus on the basics of Bitcoin and the crypto blockchain. This will include discussions on privacy and scalability within Bitcoin and blockchain technology, and an introduction to multi-party computation.3) To explore general cryptographic protocols such as Authenticated Key-Exchange (AKE) and Password-based Authenticated Key-Exchange (PAKE). The course also introduces students to zero-knowledge protocols, authentication/identification protocols, secure voting schemes, and quantum key establishment.4) To provide a solid foundation in system security and trusted computation by exploring operating system security mechanisms and policies. This includes an in-depth study on trusted execution and authenticated boot. Overall, the course aims to furnish students with a thorough understanding of information security from various angles, preparing them for future careers or research in the field.

Course learning outcomes

Upon successful completion of this Information Security course, students should be able to:1) Comprehend and apply the principles of public-key cryptography, including RSA encryption, signatures, Diffie-Hellman key exchange, and understand how to use RSA securely. They should also be able to navigate the landscape of Public-key Infrastructure, including certificates and SSL protocol.2) Understand and explain the basics of blockchain protocols, with a specific focus on Bitcoin and crypto blockchain. Students should be able to discuss issues of privacy and scalability and be familiar with multi-party computation. Demonstrate understanding of various cryptographic protocols, such as Authenticated Key-Exchange (AKE), Password-based Authenticated Key-Exchange (PAKE), zero-knowledge protocols, authentication/identification protocols, and secure voting schemes. They should also understand the basics of quantum key establishment.3) Explain and assess system security mechanisms and policies, with a particular emphasis on operating system security. Students should be able to evaluate and discuss trusted execution and authenticated boot.4) Apply the knowledge and skills learned in this course to analyze and solve problems in information security. This will prepare them for roles in the information security sector, or for further academic study in the field.5) Develop a comprehensive understanding of the current challenges and opportunities in the field of information security and be prepared to stay updated as the field evolves.

Description

1. Information security 1.1. Public-key cryptography (Jean-Sebastien Coron) 1.1.1. Introduction to public-key cryptography: RSA encryption, signatures, and DH key exchange (recap) 1.1.2. Security models in cryptography. How to encrypt and sign securely with RSA. OAEP and PSS 1.1.3. Public-key infrastructure. Certificates, SSL protocol 1.2. Blockchain protocols (Sergiu Bursuc) 1.2.1. Basics of Bitcoin and of the crypto blockchain 1.2.2. Bitcoin / blockchain privacy and scalability 1.2.3. Multi-party computation 1.3. General cryptographic protocols (Peter Ryan) 1.3.1. Authenticated key-exchange (AKE) and password-based authenticated key-exchange

(PAKE) 1.3.2. Zero-knowledge protocols 1.3.3. Authentication/identification protocols 1.3.4. Secure voting schemes 1.3.5. Quantum key establishment 1.4. System security and trusted computation (Marcus Völp) 1.4.1. Operating system security mechanisms and policies 1.4.2. Trusted execution and authenticated boot

Assessment

Assessment modality: Continuous assessmentAssessment tasksTask 1: Take-home assignment (21%)Grading scheme: 20 points (0-20) Objectives: Ability to understand the notions of public-key cryptographyAssessment rules: Students will work on their own.Assessment criteria: Produced code and report will be examinedTask 2: Take-home assignment (21%)Grading scheme: 20 points (0-20) Objectives: Understanding of bitcoin and cryptocurrenciesAssessment rules: Students will work on their own.Assessment criteria: Produced code and report will be examinedTask 3: Take-home assignment (44%)Grading scheme: 20 points (0-20) Objectives: Understanding of cryptographic protocolsAssessment rules: Students will work on their own.Assessment criteria: Produced code and report will be examinedTask 4: Take-home assignment (14%)Grading scheme: 20 points (0-20) Objectives: Understanding of system securityAssessment rules: Students will work on their own.Assessment criteria: Produced code and report will be examinedTask 5: Take-home assignment – Retake (100%)Grading scheme: 20 points (0-20) Objectives: Understanding of the concepts of the course.Assessment rules: Students will work on their own.Assessment criteria: Produced code and report will be examined.

Note

Syllabus

☒Yes ☐No

Remarks:

On Moodle

Literature list

☐Yes ☒No

Remarks:

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=4192

FORMAL METHODS

Objectives

The course starts with a gentle introduction to Software Engineering emphasizing the current state of the profession concerning the constant presence of errors that cause malfunctions in a variety of software systems. It shows how formal methods can help to overcome this situation.After that, the B formal notation is introduced by means of several practical, real-world examples. This allows to introduce set theory, binary relations, predicate calculus and how state machines can be described by using these mathematical tools. It is also shown that state machines are a suitable way of describing a large class of software systems. In particular, it is explained the concept of state invariant and how it can be proved or disproved (informally).As a third step, a tool named Set Log is introduced, and it is emphasised the fact that B specifications can be easily translated into Set Log. Then, it is shown how the translated specification becomes both a formula and a program. As a program, it can be executed thus allowing engineers to have a prototype of the system. Besides, it is shown how Set Log can automatically and formally prove or disprove state invariants.Therefore, the course will show how the mere fact of writing a formal specification enables a chain of verification activities that help to increase the quality of the final product.

Course learning outcomes

At the end of the course the student will be able to:•Know the rationale behind formal methods and what they are used for. •Write formal specifications using formal languages.•Specify state-based abstract machines. •Know, understand, and specify invariants.•Know, understand, and specify pre and post conditions.•Perform formal verification.

Description

The following topics are covered in the course:• Introduction to formal methods• The B-Method• The B formal language• Binary relations, functions, and sequences in set theory • The {log} (read as setlog) tool.• The {log} language.• Translation of a B machine into {log}• Automated proofs with {log}

Assessment

Assessment modality: Combined assessmentAssessment tasksTask 1: Written exam (30%)Grading scheme: 20 points (0-20) Objectives: Assessing the students' understanding of how to specify requirements using one of the formal languages presented in the course and its associated tool support. Assessment rules: The student uses his/her personal computer to write the specification. Any resource is available. The specification is electronically submitted via the official learning management system used at the institution. Late submissions as well as any damage or corrupted submitted file will be graded with ZERO. Assessment criteria: The provided specification must be correct. Correctness means that it passes the built-in controls of the tool used to write the specification (syntax, type-checking), but also it describes the requested requirements respecting the specification style taught during the course. Task 2: Written exam (70%)Grading scheme: 20 points (0-20) Objectives: Idem to Task 1Assessment rules: Idem to Task 1Assessment criteria: Idem to Task 1Task 3: Retake (100%)Objectives: A student who has failed the course is allowed to sit to the next exam call. The objectives are the

same as Task 2.Grading scheme: 20 points (0-20) Assessment rules: Idem to Task 2.Assessment criteria: Idem to Task 2.
Note
Syllabus

☒Yes ☐No

Remarks:

Available on Moodle.

Literature list

☒Yes ☐No

Remarks:

Schneider, Steve. The B-Method: an Introduction. New York NY: Palgrave Macmillan, 2001. Print.

(Optional complementary reading) Abrial, J., Hoare, A., & Chapron, P. (1996). The B-Book: Assigning Programs to Meanings. Cambridge. Available at the LLC (hard copy and online).

Lecture notes on B and {Log}. Available on the Moodle's course's page on due time.

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=4195

INTELLIGENT SYSTEMS 2

Objectives

Understand how logics are used for knowledge representation and reasoning, to model uncertainty, multi-agent systems, defeasibility, or causality.

Course learning outcomes

At the end of the course, the student will be able to:•Know and remember what classical, modal, and non-monotonic logics are.•Represent facts using these logics.•Reason using these logics.

Description

Part I: Logical foundations.• Propositional logic• First-order logic• Modal logicPart II: Modelling agents.• Belief and preference states • Belief dynamics• Action logics• Multi-agent systemsPart III: Non-monotonic reasoning.• Classical approaches• Valuation-based approaches • Formal ArgumentationPart IV: Probabilistic reasoning.• Probabilistic logics• Inductive probabilistic inference • Causal reasoningPart V: Practical knowledge representation.• Applications to Knowledge Representation

Assessment

Assessment modality: Combined assessmentAssessment tasksTask 1: Written exam (70%)Grading scheme: 20 points (0-20) Objectives: Prove that the logical concepts and methods have been understood and can be applied for problem-solving.Assessment rules: Open book exam (printed course materials are allowed). Assessment criteria: Correctness of the answers to questions.Task 2: Take-home assignment (30%)Grading scheme: 20 points (0-20) Objectives: Weekly test to prove that the concepts and methods are being understood.Assessment rules: Provide answers in LaTeX or handwritten text.Assessment criteria: Correctness of the answers to questions.Task 3: Written exam – RETAKE (100%)Grading scheme: 20 points (0-20) Objectives: Prove that the logical concepts and methods have been understood and can be applied for problem-solving.Assessment rules: Open book exam (printed course materials are allowed). Assessment criteria: Correctness of the answers to questions.

Note

Syllabus

☒Yes ☐No

Remarks:

Available on the course’s Moodle page.

Literature list

☒Yes ☐No

Remarks:

Relevant texts will be indicated for the individual topics.

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=3918

USER CENTERED DESIGN
Objectives
The course provides numerous insights into user-centred design methods and tools such as:

(1) choosing and applying design methods,

(2) reporting the findings, and

(3) understanding and interpreting the findings. User-Centred Design provides hands-on practice on improving interfaces and products with regards to users' needs.

Course learning outcomes
Gaining experience with user-centred design methods though hands-on exercises

Developing user-centred thinking skills
Description
1. Introducing User-Centred System Design 2. Understanding Users & Context 3. Interviewing Users 4. Structuring Findings 5. User Story Mapping 6. Storyboarding 7. Brainstorming and Prototyping 8. User Testing 9. Exploring User Interface 10. Usability and User Experience Scales 11. Effective Reporting 12. Student Presentations 1 13. Student Presentations 2 14. Wrap-up
Assessment
Assessment modality: Combined assessmentAssessment tasks Task 1: Project Presentation (100%)Grading scheme: 20 points (0-20)Objectives: Students will learn how to design and evaluate a user-centered interface, and how to effectively communicate the results of their project via presentation. Assessment rules: Throughout the course, students will work individually on a design research project. Each student must show their week-to-week homework during the course to demonstrate how they applied the methods presented during the course, what worked well and what did not, and iteratively improve their designs. Assessment criteria: The final evaluation will be based on each student's presentation of the design project undertaken during the semester. If the delivered homework assignments do not meet the expected learning objective, the student will be asked to rework their project. The attainment of learning goals will be evaluated through the project presentation based on 3 criteria: application of the introduced methods insights, lessons learned & critical thinking. presentation quality Task 2: RETAKE (100%) Grading scheme: 20 points (0-20)Objectives: Students will learn how to design and evaluate a user-centered interface, and how to effectively communicate the results of their project via presentation. Assessment rules: Video Presentation of an Interface Design and Evaluation Project (20 minutes). Assessment criteria: The attainment of learning goals will be evaluated through the project presentation based on 3 criteria: application of the introduced methods insights, lessons learned & critical thinkingpresentation quality
Note
Syllabus

☒Yes ☐No

Remarks:

Available on Moodle

Literature list

☒Yes ☐No

Remarks:

Reference textbook:

Cooper, A., Reimann, R., Cronin, D., & Noessel, C. (2014). About face: the essentials of interaction design. John Wiley & Sons.

The Encyclopaedia of Human-Computer Interaction 2nd Edition – Available Online: https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed

Recommended books:

Norman, D. (2013). The design of everyday things: Revised and expanded edition. Basic books.

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=4332

Other, please specify:

Class attendance is not required but highly encouraged.

BACHELOR SEMESTER PROJECT 6

Objectives

During a standard BSP, students discover research and development domains, produce concrete artefacts related to computer science knowledge areas covered in the BICS, collaborate with UL employees in a project context, learn new technologies related to computer science, learn new knowledge related to computer science, apply the scientific and technical knowledge learned during the BICS, apply the primary and secondary languages knowledge learned during the BICS.The course content and learning objectives are consistent with the BiCS programme content that the student is following.

Course learning outcomes

At the end of the BSP, the student should have acquired specific learning outcomes defined on an individual basis between the academic advisor(s) and him/herself. The learning outcomes should be agreed upon at the beginning of each project and be aligned with the general BiCS curriculum.

Description

Specific course content is defined on an individual basis between the academic advisor(s) and the student for each project.

Assessment

Assessment modality: Combined assessmentAssessment tasks

Task 1: Final report and in-person presentation (100%) Grading scheme: 20 points (0-20)Objectives: Assess the quality and quantity of work done during the semester. Assessment rules: Time constrained submissions. Assessment criteria: The detailed criteria are specific to each project subject and applied on the following weighted deliverables: technical & scientific quality of the final report, primary language (60%) technical & scientific quality of the final in-person presentation, primary language (30%) technical & scientific quality of the final report, secondary language summary (5%) technical & scientific quality of the final in-person presentation, secondary language summary (5%) Task 2: RETAKE (100%) Grading scheme: 20 points (0-20)Objectives: Option to repeat a previously failed project (with a final assessment of <10 points) with the same topic and under the same supervisor. Assessment rules: The same deliverables as for the previously failed project must be resubmitted under the given time constraints. Assessment criteria: Same as for Task 1.

Note

Syllabus

☒Yes ☐No

Remarks:

Accessible from the Moodle course page

Literature list

☒Yes ☐No

Remarks:

Accessible from the Moodle course page

Moodle page

☒Yes ☐No

Remarks:

https://moodle.uni.lu/course/view.php?id=4197