Certificate of Attendance

Department of Computer Science
BSc in Computer Science

Jón Agnar Stefánsson
Soc.sec: **080593-2449**

Course: T-109-INTO	Introduction to Computer Science	
Semester: 20133	**ECTS:** 6	**Grade:** 8,0

Description

This course covers fundamental concepts and foundations of Computer Science (CS) and its roles and responsibilities in society. It discusses algorithms in detail, gives examples of classic algorithms, and students get exercise in writing their own algorithms. The basics in technial writing will be covered, and students will learn about how to properly cite other sources. Furthermore, the course discusses issues such as copyright, intellectual property, privacy and security, as well as the history and the future of computing. Finally, students will get some experience in working with various tools, techniques, languages and more, such as databases, scripting languages etc.

Course objectives:

Knownledge

- know what computer science is, its main ideas and theories, and how they influence the society
- know the history of computing, and its main sections
- know the basics of how operating systems work
- know the main elements of UI programs, and know how to write a simple UI program
- know the basics of the internet
- know the major programming languages and their types
- know what to consider when writing reports
- can discuss issues such as copyright, intellectual property, privacy and security

Skills

- are capable of writing simple algorithms
- know how to use the command line
- can write reports
- can write simple programs to control small machines with Arduino
- can create a small database, fetch data from it, and insert into it, using SQL

Competence

- can apply computational thinking to real-life problems

Assessment methods:

The final grade will be calculated like this:

- Written final exam: 60%. Passing the final exam is required in order to pass the course.
- Weekly online exams: 10% (10 best grades, 1% each)
- Weekly practical exams: 30%, where each assignment is either 2% or 4% of the final grade (see syllabus and/or introduction slides)

Course: T-107-TOLH	Computer Architecture	
Semester: 20133	**ECTS:** 6	**Grade:** 6,5

Description

- In this course, you will learn how the fundamental operations of a computer. You will learn how and why the CPU -- the machine's brain -- works, how it interprets data, and how it manipulates types. You will learn basic proficiency with the UNIX operating system and how to write simple C programs.
- How to work on Linux in bash
- What are programs and how they run
- How to write basic programs in C
- Data representation
- Bit operations
- Integer and floating point representations
- Intel x86 assembly code using AT & T syntax

Course objectives:

On completion of the course the students should:

- gain insight into the fundamental hardware building blocks of the computer
- understand how data is stored and retrieved
- understand how hardware is used to perform low-level operations such as arit

Assessment methods:

To be announced.

Textbook(s):

• Computer Systems: A Programmer´s Perspective, Randal E. Bryant and David R. O´Hallaron, (2011)

Course: T-111-PROG	Programming	
Semester: 20133	**ECTS:** 6	**Grade:** 7,5

Description

This is an introductory course in computer programming using the C++ language. Fundamental programming constructs are covered, e.g. variables, types, control structures, functions and pointers, as well as built-in data structures like arrays, strings and vectors. The concept of a class is introduced and how it supports encapsulation and information hiding in the context of object-oriented programming. Students learn to use both an Integrated Development Environment (IDE) and command prompt mechanisms for development and compilation.

Course objectives:

Knownledge

- Be able to describe what is involved in the concepts of encapsulation, information hiding and abstract data type .
- Be able to describe how classes support the above concepts.
- Understand the difference between a declaration and implementation.

Skills

- Use an Integrated Development Environment (IDE) to develop and compile programs.
- Analyze, design, implement, test, debug, modify, and explain a program that uses each of the following fundamental programming constructs: variables, types, expressions and assignment, simple I/O, standard conditional and iterative structures, and functions.
- Choose appropriate conditional and iteration constructs for a given programming task.
- Apply a top-down design to break the program up into smaller units.
- use call by value and call by reference parameters.
- Write applications using pointer and dynamic arrays.
- use polymorpism in functions.
- be able to design, implement, test, debug, change and explain object oriented applications.

Competence

- Designe and develop a program for problems described in general terms.

Assessment methods:

Assessment in this course is as follows:

Homework: 15%
Section Projects: 7% (7 of 9 best apply to private)
Two Partial Tests in the semester 16% - If any part test is not passed the student is given the opportunity to repeat it a week later (except for the 2 test is repeated in Week 9). Also given the opportunity to repeat the test at the end of another. It is expected that trainees will take place of the test on a computer, but distance learners who are not able to meet the test of taking exams on paper.
Final exam: 62%. Achieve a 4.8 minimum to pass the course.
Disclaimer: As part of the test computers have not been implemented before and reasonable attempt to discuss the right is reserved to change the assessment later. If done carefully, it will be presented on the website of the course and lecture.

All projects must be submitted at the latest. 23:59 on the due date.

Textbook(s):

• PROBLEM SOLVING WITH C++, SAVITCH, WALTER, (2011)

Course: T-117-STR1	Discrete Mathematics I	
Semester: 20133	**ECTS:** 6	**Grade:** 7,0

Description

The main material in this course consists of various aspects of mathematics that are basic to an understanding of the fundamentals of Computer Science. Various topics are discussed and their relevance to practical issues in Computer Science demonstrated. Topics covered include: logic and set theory, functions, relations, matrices, mathematical induction, counting techniques and graph theory.

Course objectives:

On completing the course, students should:

Assessment methods:

Assignments and final exam.

Textbook(s):

• Discrete mathematics and Its Applications, Kenneth H. Rosen, ( )

Course: T-110-VERK	Problem Solving	
Semester: 20133	**ECTS:** 6	**Grade:** 7,0

Description

In the course the students develop skills in practical use of fundamental computer science, including programming, discrete mathematics and logical devices. Sophisticated working pratice will be emphazised and students will develop their skills in technical writing and presentation as well as ways to solve problems. All assignments are solved in-class, where students obtain advice from teachers and teaching assistants.

Course objectives:

Knowledge:

- Students should demonstrate basic command and understanding of various fundamental computer science concepts, such as game theory, state machines, grammars, regular expressions, recursion, dynamic programming and concurrency.

Skills:

Students should develop skills in:

- General problem solving.
- Technical writing.
- Technical talks.
- Groupwork.
- Using LaTeX.
- Solving problems programmatically.

Competence:

- Students should be able to solve problems, either by themselves or in a group, and communicate their solutions either in written form or verbally.

Assessment methods:

Course evaluation is as follows:

- 40% - written assignments
  - 8 best assignments out of 10
- 30% - electronic assignment
  - 11 best assignments out of 13
- 5% - student lecture
- 5% - peer review activity
- 20% - oral exam
  - **Students must pass the oral exam to pass the course**

Note that you are only allowed to miss two written assignments and two electronic assignments.

Course: T-112-OPDT	Open Exercise classes	
Semester: 20133	**ECTS:** 0	**Grade:** Passed

Course: T-216-GHOH	Software Requirements and Design	
Semester: 20141	**ECTS:** 6	**Grade:** 7,5

Description

This course studies methods for specifying, organizing and working with software requirements and software design. The needs of the user and the structure of systems are modelled and co-operation with the user is practiced. The students learn different design methods for designing systems and the user interface. The main focus is on practicing different methods for requirements specification, analysis, design and testing in the early phases of software development.

Course objectives:

On completing the course, students should be able to:

Apply key elements and common methods for elicitation and analysis to produce a set of software requirements and user requirements for a medium-sized software system. Describe the functional and non functional requirements for the system Conduct a review of a software requirements description using best practices to determine the quality of the description Use a common, non-formal method to model the requirements for a medium-size software system.

Design the user interface of the system using both paper and software prototypes Identify several fundamental principles for effective GUI design and summarize the major GUI guidelines and standards Evaluate prototypes with users to adjust the requirements for the system Discuss the properties of good software design.

Discuss in what ways might the design of a computer system or application succeed or fail in terms of respecting human diversity. Compare and contrast object-oriented analysis and design with structured analysis and design. Evaluate the quality of multiple software designs based on key design principles and concepts. Select and apply appropriate design patterns in the construction of a software application.

Create and specify the software design for a medium-size software product using a software requirement specification, an accepted program design methodology (e.g., structured or object-oriented), and appropriate design notation. Conduct a software design review using appropriate guidelines. Evaluate a software design at the component level. Evaluate a software design from the perspective of reuse.

Assessment methods:

- Final exam (must achieve a grade higher than 4,75 out of 10 in order to pass the course): 50%
- Hand-in assignments (4 at 7,5% each): 30%
- Online exams (2 out of 3 at 2,5% each): 5%
- Weekly assignments (must hand in all but two): 5%

Textbook(s):

• Introduction to Systems Analysis & Design. An Agile, Iterative Approach, Satzinger, Jackson, Burd, (2012)

Course: T-419-STR2	Discrete Mathematics II	
Semester: 20141	**ECTS:** 6	**Grade:** 8,0

Description

This course is the follow-up of Discrete Mathematics I. It covers the basics of three topics in discrete mathematics that are of fundamental importance in the theory and practice of computer science:

- grammars and finite automata as models of languages and computation, respectively;
- linear algebra and its applications in computer science, with emphasis on solving systems of linear equations using Gaussian elimination. matrix algebra and matrix transformations, operations on vectors (scalar multiplication, dot product and cross product) and equations for lines and planes;
- discrete probability, with focus on assigning (conditional) probabilities, Bayes´ Theorem and expectation.

Course objectives:

Upon completion of the course, the students should:

- Be able to formulate inductive definitions of discrete structures, such as strings and trees, and construct proofs by structural induction over those structures.
- Be able to argue whether an infinite set is countable or uncountable.
- Be able to design grammars generating some simple languages and finite automata accepting them.
- Be able to write regular expressions denoting some simple regular languages.
- Be aware that the halting problem is algorithmically unsolvable.
- Be able to solve systems of linear equations using Gaussian elimination.
- Be able to use matrix algebra and geometric transformations in computer graphics.
- Be able to use operations on vectors (scalar multiplication, dot product and cross product) and to write equations for lines and planes.
- Be able to assign probabilities to events over finite probability spaces.
- Be able to calculate conditional probabilities.
- Be able to apply Bayes´ Theorem to estimate probabilities based on partial evidence.
- Be able to calculate the expected value of a random variable.

Assessment methods:

There will be five exercise sheets (each worth 2% of the final mark for the course), two test assignments (each worth 5% of the final mark) and a final, written exam (contributing 80% of the final grade). Each student must pass the final exam with a grade of 5 in order to pass the course.

Bonus points up to a maximum of 0.5 will be given for course participation to those who pass the final exam for the course. Course participation will be measured using activity on Piazza. Only good questions and answers count.

Textbook(s):

• Discrete mathematics and Its Applications, Kenneth H. Rosen, (2013)

Course: T-213-VEFF	Web-Programming	
Semester: 20141	**ECTS:** 6	**Grade:** 8,0

Description

This course will introduce the basics of web applications and design patterns. The focus will be on the protocols and standards that are used in all server-side web frameworks, and OOP concepts used when implementing design patterns. Students will also learn how to improve security in websites, how to handle errors gracefully, how to write and consume web services, as well as learn about the standards and best practises in their implementation. Furthermore, students will learn how to connect to a database and other types of data storage in application code. Finally, the future of web applications and web standards will be discussed.

Course objectives:

- be able to create a small to medium sized web application using one of the major serverside web frameworks
- understand web protocols and web standards
- be familar with the basics of client-side programming in web applications
- understand how to build web services, and how to consume them
- understand the basics of design patterns and know when to apply them
- apply OOP principles, like inheritance and interfaces, in the context of web applications
- know how to connect to various types of data storages in application code
- be familar with best practises in graceful error handling

Assessment methods:

Hand-in assignments 35%

- Assignment 1 (7%), due in week 3
- Assignment 2 (7%), due in week 5
- Assignment 3 (7%), due in week 7
- Assignment 4 (7%), due in week 9
- Assignment 5 (7%), due in week 11

Lab assignments, to be completed in practical session, 5%

- 7 assignments, 1% each, best 5 of 7

Weekly online exams 10%

- 12 online exams, 1% each, best 10 of 12

Final exam 50%

- Will be written
- Will be both in Icelandic and English
- To pass the course you must pass the final exam (with 4,75 or higher)
- No helping material allowed in the final exam

Textbook(s):

• Web programming, Daníel B. Sigurgeirsson, (2011)

• PRO C# 2010 & THE .NET 4 PLATFORM, Andrew Troelsen, (2010)

Course: T-201-GSKI	Data Structures	
Semester: 20141	**ECTS:** 6	**Grade:** 7,0

Description

This course discusses various data structures, like linked lists, stacks, queues, trees and hash tables. Recursive programming and sorting algorithms are also discussed . At the same time, emphasis is put on abstract data types, object-oriented programming and exception handling. The programming language C++ is used in the course.

Course objectives:

On completing the course, students should be able to:

- Write programs that use each of the following data structures: multi-dimensional arrays, linked lists, stacks, queues, trees, and hash tables.
- Choose the appropriate data structure for modeling a given problem.
- Identify the base case and the general case of a recursively defined problem.
- Implement, test, and debug simple recursive functions.
- Compare iterative and recursive solutions for elementary problems such as factorial.
- Apply sequential search, binary search and sorting algorithms in a creative manner or in new circumstances.
- Describe the ideas underlying the notion of abstract data types and the differences between their specification and implementation.
- Be able to use abstract data types by only having access to their specification.
- Justify and describe the philosophy of object-oriented design and the concepts of encapsulation, inheritance and polymorphism.
- Design, implement, test, and debug programs in an object-oriented programming language.
- Develop code that responds to exception conditions raised during execution.

Assessment methods:

The course assessment in this course is threefold:

Main assignments: 25% (5 in total)

Lab assignments: 10% (8 out of 11 count towards the final grade)

Optional assignments 5% (count towards the assignment grade)

Exam: 65%

5,0 is the minimum in order to pass the course.

All assignments are due at 23:59 on the return date. Late returns bear the penalty of 10% for each day (including weekends and holidays). Assignments are not accepted if they are handed in more than two days late.   All assignments must be handed in to Mooshak. An assignment will only be graded if it has been submitted to Mooshak and it compiles (i.e. Mooshak does not give a Compile Time Error).

Textbook(s):

• Problem Solving with C++, Walter Savitch, (2012)

Course: T-205-VERK	Practical Project	
Semester: 20141	**ECTS:** 6	**Grade:** 9,0

Description

Students do a realistic software development project in a multi-user environment. The aim of the project is to specify, design and implement usable software involving a relational database. Students work in small teams under the guidance of a supervisor. At the end of the project students submit the software and documentation and undergo an oral exam.

Course objectives:

On completion of the course students should:
• Have created a system that uses a relational database in a multi-user environment
• Have used standard development methods for software development
• Be familiar with programming in a window environment
• Be familiar with web-based programming

Assessment methods:

Appraisal of documentation and software produced.

Course: T-301-REIR	Algorithms	
Semester: 20143	**ECTS:** 6	**Grade:** 7,0

Description

This course introduces the most important types of algorithms and data structures in use today. Emphasis is placed on algorithms for sorting, searching and graphs. The focus is on developing implementations, analyze them or evaluate empirically, and assess how useful they can be in actual situations.

Course objectives:

Knowledge
- Describe the efficiency of the major algorithms for sorting, searching and hashing.
- Describe the problem with exponential growth of brute-force solutions and its consequences
- Give examples of the applications of graphs, trees and symbol tables

Skills
- Skilgreint reiknileg verkefni formlega út frá almennri lýsingu
- Beitt mismunandi rakningaraðferðum á tré og net
- Rakið framkvæmd aðgerða á klassískar gagnagrindur: hrúgur, tvíleitartré, rauð-svört tré, union-find.
- Leyst verkefni með grundvallarreikniritum fyrir net, svo sem dýpt-fyrst og breidd-fyrst leitun, gagnvirk lokun (*transitive closure*), grannröðun (*topological sort*), og reikniritum fyrir stystu leiðir og minnstu spanntré.
- Metið áhrif mismunandi útfærslna hugrænna gagnataga (ADT) á tímaflækju reiknirita.
- Beitt „big-O“, omega og þeta ritihætti til að gefa efri, neðri og þétt mörk á tíma- og plássflæku reiknrita í aðfellu.
- Beitt vísindalegri aðferð við mælingar á reikniritum.
- Útfært stofnrænar (*generic*) gagnagrindur og beitt þeim á mismunandi gögn.
- Greint tímaflækju deila-og-drottna reiknirita með lausn mismunajafna.

Competence
- Metið reiknirit, valið á milli mögulegra lausnaraðferða, rökstutt það val, og útfært í forritum.
- Þróað lausnaraðferð byggða á deila-og-drotta, gráðugri reglu eða kvikri bestun (*dynamic programming*).

Assessment methods:

To be determined.

Textbook(s):

• Algorithms, Sedgewick, Wayne, (2011)

Course: T-317-CAST	Calculus and Statistics	
Semester: 20143	**ECTS:** 6	**Grade:** 7,0

Description

The course is divided into two parts, calculus in the first half and statistics in the second half.

Calculus – Limits, differentiation and integration of functions of one variable.

Statistics – An introduction to probability and statistics: - Discrete probability distributions, in particular the binomial distribution. - Continuous probability distributions with the main emphasis on the normal distribution. - Confidence intervals and hypothesis testing. - Correlation and linear regression.

Course objectives:

Know basic properties of functions of one variable, such as polynomial, rational, logarithmic, exponential and trigonometric functions. Be familiar with the basic concepts of calculus, such as continuity and differentiability. Know the derivatives of common functions of one variable, such as polynomial, rational, logarithmic, exponential and trigonometric functions.

Understand integration and know integrals for common functions of one variable. Be familiar with integration by substitution and integration by parts. Know what a differential equation is and have seen some examples of differential equations. Be familiar with discrete probability and some basic methods for its calculation, in particular permutations and combinations. Be familiar with discrete probability distributions, in particular the binomial distribution, and know how to compute their expected value and standard deviation.

Know continuous probability distributions, in particular the normal distribution and the t-distribution. Be familiar with confidence intervals and hypothesis testing. Be familiar with correlation and regression. Be able to differentiate functions of one variable, such as polynomials, rational, logarithmic, exponential and trigonometric functions. Be able to use differentiation to sketch the graphs of various functions of one variable.

Be able to integrate various functions of one variable, for instance using integration by substitution or integration by parts. Be able to solve various practical problems by constructing a function and using differentiation to find its maximum or minimum value. Be able to solve very simple differential equations, for example for exponential growth or exponential decay.

Be able to calculate discrete probability using techniques such as permutations and combinations. Be able to calculate expected value and standard deviation for discrete probability distributions, such as the binomial distribution. Be able to compute probabilities for continuous variables, using for example the normal distribution or the t-distribution.

Be able to compute confidence intervals and test hypotheses. Be able to compute the correlation coefficient and find a regression line. Be able to analyze various problems and apply the methods of the calculus of one variable to solve them. Be able to apply hypothesis testing to analyze sets of measured data.

Assessment methods:

Textbook(s):

• Elementary Statistics - a step by step approach, Bluman, ( )

• Calculus Early Trancedental Functions, Robert T. Smith og Roland B. Minton, ( )

Course: T-202-GAG1	Databases	
Semester: 20143	**ECTS:** 6	**Grade:** 8,0

Description

The course is a hands-on introduction to information management in general and relational database management in particular, covering the following topics: the role and function of database management systems; the relational database model, including relational concepts and relational query languages; data modeling using the ER model and its conversion into a relational database schema; all major aspects of the SQL language, covered in detail, including DDL, DML, complex queries, views, procedures, triggers and transactions; transaction and administration concepts; and, finally, a brief discussion of alternative data models and approaches, such as unstructured databases, information retrieval and "big data".

Course objectives:

Knownledge

- Discuss structured and unstructured databases in social and organizational context.
- Describe concepts and measures related to reliability, scalability, efficiency and effectiveness.
- Describe major components and functions of database management systems.
- Describe and compare common data models.
- Describe fundamental principles of the relational model.
- Describe fundamental transaction concepts.
- Describe basic database administration functions.
- Discuss concepts and techniques for unstructured data and information retrieval.
- Discuss major approaches to storing and processing large volumes of data.

Skills

- Write SQL commands to create a complete relational database.
- Write SQL commands to insert, delete, and modify data.
- Write simple and complex SQL queries to retrieve data, including joins, aggregates, sub-queries, nested sub-queries, and division.
- Write simple database views, stored procedures, triggers and transactions.
- Write queries in relational algebra and tuple relational calculus.

Competence

- Model data requirements and constraints using the ER-model
- Convert an ER-model into a corresponding relational schema.
- Normalize a relational schema.
- Select and create the appropriate indices for simple database queries and constraints.

Assessment methods:

The assessment will be as follows:

Weekly assignments10%: 12 assignments will be given and the best 10 count towards the grade

Group projects40%: 5 projects will be assigned and the best 4 count towards the grade

Final examination50%: According to RU regulations, passing the exam is required to pass the course

Textbook(s):

• Database Management Systems, Ramakrishnan - Gehrke, ( )

Course: I-406-IERP	Introduction to ERP Systems	
Semester: 20143	**ECTS:** 6	**Grade:** 10,0

Description

The largest investment for companies is in business systems (Enterprise Resource Planning; ERP). ERP actually is an alias for the interaction of processes and technology operations, including planning, procurement and product management, human resources, finances, inventory and sales. In recent years there has been considerable development in this sector and the systems have become much more versatile and powerful than before. According to a recent survey from Gartners investments in information technology will continue to increase in coming years. The main obstacle for growth in this sector relates to the lack of staff who has expertise in this area, which is a compbined knowledge of the technical details of information technology and the main conversion processes. The largest players in this market are SAP, Oracle and Microsoft and the work environment is global. This course seeks to create a solid foundation for anyone who wants to study the function and development of ERP systems. We will look into different ways of implementation and the impact that ERP systems have on business operations. This course will be based on Microsoft Dynamics NAV in teaching. By the end of the course students need to be familiar with and understand the functionality and key features of ERP systems. Students will also get the chance to take a look "under the hood" and an opportunity to work in a technical environment.

Course objectives:

Knowledge
- Be able to describe the functions and use of ERP systems and their development over time
- Be able to define key components of ERP systems and describe their context

Skills
- Be able to work with the main system components in Microsoft Dynamic NAV
- Be able to program simple functionality with the ERP system

Competence
- Be able to design processes within the ERP system

Assessment methods:

Assignment 1 - 30%

Assignment 2 - 30%

Assignment 3 - 40%

Course: T-303-HUGB	Software Engineering	
Semester: 20143	**ECTS:** 6	**Grade:** 7,5

Description

The main topic of this course is the software development life-cycle and working methods associated with the life cycle. We will look at the history of development models, the role of individual methods will be discussed, with an emphasis on Agile development models and practices within those models. The integration and relationship between practices will be covered, with special emphasis on how they should support an incremental delivery of value to the customer, and even more, the continuous improvement of the process an organization is using. This course covers the following: Development models, practices and supporting tools, project management, quality management and related cultural aspects, planning, cost estimates, software metrics, configuration management, testing and teamwork. We will also seek to get experienced individuals from industry to share their vision on software development with students.

Course objectives:

On completion of the course students should:

- obtain an understanding on the software process as a whole
- adopt a disciplined and exclusive working procedures for developing, running and maintaining software
- understand the importance of Software Quality Assurance
- get to know the basics for Software Project Management
- understand the purpose of Software Configuration Management

Assessment methods:

To be announced.

Textbook(s):

• The Art Of Agile Development, James Shore & Shane Warden, (2007)

• Scrum and XP from the Trenches, Henrik Kniberg, (2007)

Course: T-239-KERF	System Administration 1	
Semester: 20151	**ECTS:** 6	**Grade:** 8,0

Description

The goal of this course is to introduce students to what system administrators do and what it means to run a computer system. Different operating systems will be covered. Hardware in servers will be discussed (though not those common in terminals or home machines) and the environment that servers are kept in. Maintenance of computer operating systems will be looked at, such as how to install an operating system, how it starts, etc. Discussions of measurement to find bottlenecks. Security issues and access control of resources. Major services will be introduced, on what ports they run and what they do. For example DNS, DHCP, mail (SMTP, POP, IMAP), HTTP, FTP, etc. Data backup will be discussed. There will also be discussions about maximizing uptime, load-balancing and virtual machines.

Course objectives:

Knowledge: Know how Windows and Linux operating systems launch, and what services are related to the process. Know the basic concepts in system rooms. Know different ways to ensure the operational safety of backups and sufficient supply of hardware or services. Be able to identify the differences between different RAID methods. Be able to monitor computer systems in operation. Be able to identify how the disk is partitioned and how file systems are designed. Be able to identify the difference between NAS and SAN. Be familiar with services such as AD, Kerberos and LDAP. Have gained knowledge in how computers communicate with each other through the network. Be able to automate computer systems and their operation. Skills: Be able to install Windows and Linux servers with simple services. Know the basic concepts of virtual machines and be able to set up a virtual machine. Competence: Be able to maintain the operation of Windows and Linux machines. Be able to recognize the limitations of the system rooms (net, bandwidth, cooling, power, disk space, etc.). Be able to understand the important services in operation.

Course: T-427-WEPO	Web-Programming II	
Semester: 20151	**ECTS:** 6	**Grade:** 6,5

Description

This course will focus on the client-side aspects of web programming. Subjects include:

- HTML5
- JavaScript
- Single-Page Applications (SPA) using AngularJS
- CSS
- Responsive Web Design

Course objectives:

Knownledge
- Students know the latest standards in HTML, CSS and JavaScript
- Students know about common client-side libraries, and know how to use them

Skills
- Students have gained the ability to write applications using the latest client-side technology in web programming
- Students have gained the ability to write JavaScript code

Competence
- Students can analyze what client-side techniques are best suited to solve various problems, and what pros/cons common libraries have

Assessment methods:

The assessment of the course is as follows:

- There are 4 assignments during the semester, which are total 60% of the final grade.
- There will be an online exam each week, 10% of the final grade (i.e. 10 best out of 12 exams)
- The final exam is 30% of the final grade

In order to pass the course, students must pass the final exam.

Textbook(s):

• Javascript: The Good Parts, Douglas Crockford, (2008)

• HTML5 for Web Designers, Jeremy Keith, (2011)

• Responsive Web Design, Ethan Marcotte, ( )

• Web API Design: Crafting Interfaces that Developers Love, Brian Mulloy, (2012)

• Maintainable JavaScript, Nicholas C. Zakas, (2012)

• AngularJS in 60 minutes, Dan Wahlin, (2013)

Course: T-501-FMAL	Programming Languages	
Semester: 20151	**ECTS:** 6	**Grade:** 7,5

Description

The evolution of programming languages is an important factor in computer science. The course describes this evolution from the first programming languages to the more recent languages. Different types of programming languages are discussed and their characteristics compared. Programming languages syntax is introduced as well as Backus-Naur Form (BNF). Main characteristics of imperative languages are examined, particularly regarding to scope rules and procedure activations. Focus is on the Smalltalk programming environment while discussing object-oriented languages. The constructs of functional programming languages are examined with emphasis on Lambda calculus and the ML language. Logic programming is introduced and the Prolog language is specifically analyzed. Students are introduced to the design and syntax of the above languages and experiment with several programming projects using some of these languages.

Course objectives:

Course objective is that students:

- Know the history of language developments.
- Be able to use BNF to describe programming language syntax.
- Have a thorough understanding of the main characteristics of imperative languages.
- Be familiar with the Smalltalk object-oriented programming environment.
- Know the fundamental concepts of functional programming languages and Lambda calculus.
- Be familiar with the Prolog programming language paradigm.

Assessment methods:

The course assessment is twofold:

1. Assignments: 40% (3 programming assignments, 2 problem assignments, 8% for each assignment)
2. Final exam: 60%. A minimum of 5,0 is needed to pass the course.

All assignments are due at 23:59 on the return date. Late returns bear the penalty of 10% for each day (including weekends and holidays). Assignments are not accepted if they are handed in more than two days late.

Textbook(s):

• Concepts of Programming Languages, Robert W. Sebesta, (2010)

• Programming Languages: Principles and Paradigms, M. Gabbrielli & S. Martini, (2010)

Course: T-215-STY1	Operating Systems	
Semester: 20151	**ECTS:** 6	**Grade:** 8,5

Description

All the basic issues of operating systems will be covered: Processes, threads, process comminication, deadlocks, scheduling, memory management, virtual memory, I/O, filesystems, access control and security. Examples will be taken from Unix/Linux and Windows 2000 operating systems.

Course objectives:

Knowledge
- Explain the objectives and functions of modern operating systems.
- Explain dynamic memory allocation in modern operating systems.
- Describe the need for concurrency within the framework of an operating system.
- Explain the memory hierarchy and cost-performance trade offs
- Describe the difference between processes and threads.
- Discuss the need for a hardware cache, as well as common algorithms and optimizations used to implement it
- Explain signal handling within UNIX -based operating systems.
- Explain conditions that lead to deadlock.
- Compare and contrast the common algorithms used for both preemptive and non- preemptive scheduling of tasks in operating systems, such as priority, performance comparison, and fair-share schemes.
- Explain the concept of virtual memory and how it is realized in hardware and software.
- Summarize the principles of virtual memory as applied to caching, paging, and segmentation.
- Compare and contrast paging and segmentation techniques

Skills
- Disassemble, trace and perform rudimentary debugging of programs written in Intel x86 assembly.
- Boot an operating system using a simulator.
- Write a buffer-overflow exploit

Competence
- Write working C code that interacts with standard C libraries and the operating system kernel directly.
- Write a multi-threaded multi-tenant service using semaphores and mutexes.
- Write a primitive command shell for UNIX-based operating systems

Assessment methods:

15%: 3 problem sets, this could change, please see anouncements

60%: 5 Hands-on practical assignments (labs)

25%: A closed-book final exam on which 5.0/10.0 required to pass the course.

There are bonus points in all assignments and final exam so that you can get an 11. Please keep in mind that you will not be able to get a higher final grade than 10.

Textbook(s):

• Computer Systems: A Programmer´s Perspective, Randal E. Bryant and David R. O´Hallaron, (2011)

Course: T-615-SVER	Independent Study	
Semester: 20151	**ECTS:** 6	**Grade:** 9,0

Description

Students can choose to complete a 3 credit independent project during the final semester. This course is comparable to the B.S. Essay (Thesis) but the main difference is twofold. The independent project can be a collective project for more than one student and the projects results are submitted as a project report rather than a thesis. The project is intended to develop and train new skills for the student(s) involved and therefore it should not be a typical software design project The project is done under the guidance of a supervisor but students are expected to work independently without continual supervision. Students should submit a proposal for the thesis. After the proposal has been approved by the department a supervisor is appointed.

Course objectives:

Course objectives: On completion of the course students should:
• Be able to make a search on a particular problem in Computer Science and its solution
• Be able to express the problem in a simple way and describe what solutions have been invoked
• Be able to express a reasoned conclusion about the solutions and in some cases contribute original solutions
• Be able to work independently

Assessment methods:

Hönnunarskýrsla og samskiptagildasamningur 10%
Lyftukynning x2 10%
Verkáætlun og framvinduskýrsla 15%
Lokaskýrsla og afurð 30%
Jafningjamat 20%
Lokakynning 10%
Verklag 5%

Course: T-302-HONN	Software Design and Implementation	
Semester: 20153	**ECTS:** 6	**Grade:** 7,5

Description

The main objective of the course is to show how to design and architect software enterprise solutions. The focus is on layered Internet systems with web interfaces. The course covers how to build flexible system that are easy to adapt and maintain. To realize this, the focus is on architecture and module design. Options facing architects and how to recognize important key design goals is covered. Many known design patterns are discussed and evaluated. Topics like performance and scalability of enterprise solutions are also covered. The course uses Java 2 Enterprise Edition and additional Java APIs. These topics are covered: design of database access using JDBC, Web architectures based on Model-view-controller using Java Servets and JavaServer Pages, and component programming using Enterprise JavaBeans. The course will also use open source software such as ANT, Tomcat web server and JBoss EJB server.

Course objectives:

On completion of the course students should:

- have learnt the basics of software design
- understand different software architectures and the differnet design choices
- have learnt how to use design patterns, including Model-view-controller
- implement software framework with component reuse
- be able to design and build flexible solutions
- be able to design and build high-performance and flexible solutions
- have learnt how professional practices to software development
- understand the practices of how professional software is developed in the software industry

Assessment methods:

The course has two assessments:

- Four assignments which have total value of 50%
- Final exam 50%

To pass the course you have to get minimum of 5.0 in the final exam and get a minimum of 5.0 in assignments.

Textbook(s):

• Software Architecture for Developers, Simon Brown, ( )

Course: T-514-VEFT	Web Services	
Semester: 20153	**ECTS:** 6	**Grade:** 8,5

Description

In this course students will learn to design and implement web services. The course will exam various server libraries, as well as examine the main pattern and practices in programming on the client. The course will cover the following material: server side frameworks such as ASP.NET Web API / C #, Node.js / JavaScript, REST web services (concepts, implementation) Design and pattern, Logging / Monitoring, security, Caching, Deployment and Microservices.

Course objectives:

Knownledge

- Be familiar with programming libraries on the server side
- Be familiar with REST services, and know the differences between REST and RPC services
- Be able to know the differences between services which deliver data only, and systems which render web pages server-side.

Skills

- Acquire the skills in writing backend, in at least two common environments
- Be able to set up an operable webservice

Competence

- Know which libraries will be suitable under different circumstances during server-side development
- Be able to know when to use a web service, and when to write a system which generates web pages on the server.

Assessment methods:

Final grade consist of: Written final exam: 30%. Assignments: 60%. Weekly internet exams: 10%

Course: T-409-TSAM	Computer Networks	
Semester: 20153	**ECTS:** 6	**Grade:** 8,0

Description

The course begins with a short overview of network systems and services. After introduction, the focus will be on the layers of the OSI/IETF models.

The following network layers will be studied in details:

- Application layer – WWW, HTTP, DNS, SMTP, FTP etc.
- Transport layer – UDP and TCP.
- Network layer – Link State routing and Distance vector routing, IP, IP-addresses.
- Link layer – MAC, Ethernet, Hubs and switches.

Finally, an introduction to some more specific topics such as mobile networks and network security will be given. Students will also work on the topics through programming assignments and homeworks.

Course objectives:

On completion of the course students should understand and be able to explain:

- the different layers of the ISO/OSI network protocol stack and their interaction
- the basic application layer applications and protocols such as Web, Mail and P2P applications
- transport layer protocols and mechanisms such as UDP, TCP, and congestion control
- network layer protocols and mechanisms such as forwarding and routing of messages IPv4 and IPv6 ICMP routing in the internet broadcast and multicast
- link layer protocols and mechanisms such as error detection mechanisms (parity checks, checksums, CRCs) access protocols (based on channel partitioning, random access, taking turns) link layer addressing (MAC addresses and address resolution protocol) ethernet
- the basic protocols and concepts of wireless and mobile networks characteristics of wireless networks and access protocols used Wireless LAN cellular internet access management of mobility GSM
- basic security terminology, security threads in networks and the concepts of counter measures: what is network security? symmetric and public key cryptography mechanisms to check message integrity: cryptographic hash functions, message authentication codes, and digital signatures authentication protocols security protocols in the Internet: PGP, SSL, IPsec, WEP firewalls and intrusion detection systems

In addition, the students should be able to program a simple networked application using sockets.

Assessment methods:

Homework (24%), programming assignments (30%) and final exams (46%).

There are seven homework assignments. The best six of seven will account for the grade, 4% of the final grade each, for a combined 24%. There will be three programming assignments, each for groups of two to three students. Each assignment counts 10%. The final exam will account for 46% of the overall grade for the course. Passing the final exam is necessary for passing the course.

Late submissions of assignments will be penalized by reducing the score by 15% per 24 hours or part there of that the submission is late. Submissions that are handed in later than 48 hours after the deadline will not be graded.

Textbook(s):

• Computer Networking - A Top-Down Approach, James F Kurose , Keith W Ross, (2013)

Course: T-408-IAGH	Introduction to Graphic Design	
Semester: 20153	**ECTS:** 6	**Grade:** 7,0

Description

The objective of this course is twofold. The students will learn to become self-reliant when it comes to creating the appearance of excellent CMS. The importance of communication between designers and developers will be discussed. Students will be able to communicate with developers, and may have an opportunity to design their websites. The course will cover the basics concepts of graphic design skills, acquire skills in the design and editing of the CSS/HTML descriptions.

Course objectives:

Assessment methods:

Course: T-513-LIFT	Living Technology	
Semester: 20153	**ECTS:** 6	**Grade:** 9,0

Description

Interactive experiences are common in daily life, and their ability to affect social change provides a rich foundation for innovation. This course explores the design and development of a particular kind of interactive experience: those that are driven by technology and designed to bring some benefit to society at large. The emphasis of this course will be team--based collaboration, with students from both Reykjavik University and the Iceland Academy of the Arts working together to design and develop a beneficial interactive experience over the span of three weeks. Class time will combine lectures, demonstrations, discussions, and group exercises, and lab time will be focused on team project work with instructor guidance and moderation.

Course objectives:

Upon completion of the course, students should be able to:

- Describe the formal elements of interactive experiences and the relationships between them;
- Discuss the potential benefits of interactive experiences in daily life;
- Employ focused strategies to generate ideas for novel interactive experiences;
- Communicate project ideas clearly and concisely;
- Discuss the influence of multisensory effects on people’s experiences;
- Conduct a user test to evaluate an experience;
- Describe some current directions in interactive experience research;
- Design and develop an interactive experience in a limited amount of time;
- Discuss how interactive experiences can influence people’s behaviour;
- Discuss how interactive experiences can be personalized

Assessment methods:

TBA

Course: T-404-LOKA	Final Project	
Semester: 20161	**ECTS:** 12	**Grade:** 8,5

Description

The Final Project consists of software development in collaboration with a client and users outside the university. The purpose of the final project is to give students experience of working independently on specification, design and implementation of software and to use accepted methods in the development cycle. Normally 2 to 4 students work together in a project group. While working on the project, students gain practical experience of design, programming and testing. The projects are evaluated by the project supervisor and two other internal examiners. The grade is based on evaluation at various stages of development and takes into account all aspects of the development work. The projects conclude with a public presentation.

Course objectives:

Knownledge

- Be familiar with internet programming
- Be able to identify the design of a system of average size
- Be familiar with the basics of unit testing
- Be able to define and implement a system test

Skills

- Be able to analyze, design and write medium-sized web applications based on relational databases in multi-user environments
- Gain experience working in a group to construct medium-sized systems
- Be able to write reports that describe an aspect of the development process
- Be able to use a version control system to keep track code changes
- Be able to present the final project

Competence

Assessment methods:

Appraisal of project at various stages. Appraisal of project documentation and public presentation.

Course: T-426-RUIS	RU Internship	
Semester: 20161	**ECTS:** 12	**Grade:** Passed

Description

In this course students will work on a practical project with the aim to implement and deploy the software. The course work will be according to accepted practices in the market, in a technical environment that can match the best in the business, and with the highest standards for documentation, safety and reliability of data.

Course objectives:

Be able to read code written by others and describe its functionality. Be able to improve code and documentation written by others. Be able to apply the various standards and rules set by the project. Skills: Be able to apply best practices in version management. Be able to address issues raised by code reviewers. Be able to operate in a group, which is itself a part of a larger team. Be able to present (either orally or in writing) his/her work. Be able to express what problems are encountered, and seek assistance in solving those problems. Competence: Be able to identify which parts of a project documentation need improving. Be able to identify which parts of a project codebase need improving.

Assessment methods:

Course: T-611-NYTI	New Technology	
Semester: 20161	**ECTS:** 6	**Grade:** 8,0

Description

The objective of this course is to look at new and emerging technology. Disruptive technologies from the past and their effects are covered. Technology advanced in the beginning of the 21st century are used as basis for the discussion on new technologies. The focus is on new devices such as wireless devices, mobile phones and TV and other consumer devices. The course will discuss what future trends will emerge, what standards and companies will be successful, and the effects the technology will have on society.

Certificate of Attendance

Course objectives:

On completion of the course students should:

- get insights into disruptive technolgies from history
- know about new and emerging technology and the oppertunites new technology presents
- learn about new client devices like phones and interactive TV
- have written a research paper on a technology.

Assessment methods:

GRADING

Research Report 50% (Report 40%, Draft 5%, Peer-review 5%)

Assignments 50% (six assigment each 10% with the five highest used)

There is no final exam because us hates them.

Textbook(s):

• New Technology, Ólafur Andri Ragnarsson, (2016)

Calculated average grade: 7,84

Total 180 credits of 180

Reykjavik 1.10.2024

Certified by

HÁSKÓLINN Í REYKJAVÍK REYKJAVÍK UNIVERSITY

NEMENDASKRÁ
REGISTRY

[illegible]