CS 1102: Introduction to Computing (3C, 45L)

Rationale: Computing has become an indispensable tool today in the study of physical and biological sciences. This course will motivate the student to investigate the field of computing, how it processes information, means of expressing computation, meaning of complexity, hard problems, and finally emerging domains that heavily use computing. This course is primarily intended to generate enthusiasm for studying computing.

Prerequisites: None

Intended Learning Outcomes: Upon completion of this course, students will be able to:

1. describe and review of the wide spectrum of knowledge areas that is covered by Computing;

2. explain algorithmic complexity classes and its reflection in efficiency of algorithms;

3. explain, review and illustrate the use of logical reasoning, machine learning and cryptography in information processing.

Course Content: Information: Discretisation, quantification (entropy), compression, error recovery. Floating point arithmetic: Fixed point vs. floating point representations; range, precision and accuracy in arithmetic operations, overflow and underflow. The Von Neumann computer: Basic structure, machine instruction execution. Automata principles: simple reactive systems (tea/coffee machine), string processing, Finite-state machine, Pushdown automata and Turing machine. Algorithms: *origins, a taxonomy* - solvable (polynomial) vs. undecidable; hard but easily verifiable (NP) vs. 'easy'; intractable (exponential); exact solution vs. approximate; time and space complexity; greedy vs. exhaustive search; optimal vs suboptimal; Impossibility of exhaustive search example: TSP of a 100 vertex graph has O(n!) paths; on a 10GHz machine with one path per cycle will require more than the life time of universe; Discussion of selected P-time algorithms: bubble sort, quicksort, binary search, string matching, job shop scheduling, sub set sum and matrix multiplication; Common complexity classes: P, EXP, NP with examples. Application domains overview: Information security (factorisation as a hard problem), Expert systems (Prolog based reasoning), Machine learning (pattern classification using basic ANN models), computational science (modelling of dynamical systems, cellular automata, epidemics)

Methods of Evaluation: End of semester examination (100%)

Suggested References:

Denning, P. J. Great principles of computing, MIT Press, 2015

Dromey, R. G. How to solve it by computer, PHI, 1982

Cormen, T. H., Leiserson, C. E ., Rivest, R. L ., Stein, C. Introduction to algorithms (3rd Edition), MIT Press, 2009

CS 1101: Fundamentals of Programming (3C, 30L 30P)

Rationale: This course is designed for students with little or no programming experience. It aims to provide students with an understanding of the role of computation, its involvement in solving problems and to help students, regardless of their major, feel justifiably confident of their ability to write small programs that allow them to accomplish useful goals.

Prerequisites: None

Intended Learning Outcomes: Upon completion of this course, students will be able to:

1. explain the role of computation and the way of problem solving;
2. install and setup the Python programming environment, list the steps of writing a program, describe the interpretation process;
3. identify different data types, variables, and operators; use them appropriately in programming.
4. identify the use of different flow control statements; write small programs that allow them to accomplish useful goals;
5. discuss different modules of String handling package and different building blocks of python – lists tuples etc.;
6. illustrate handling exceptions, apply event handling, and Integrate databases through python programming.

Course Content: Introduction to Programming Languages: Low-level versus high-level, General versus targeted to an application domain, Interpreted versus compiled. Python Basics: History of python, Features of python, Python Environment, Identifiers, Keywords, Comments, Basic syntax. Variables and Data Types: Expressions and Numerical Types, Conversions, Variables and Assignment, Operators. Branching and Iteration: Selection Statements, Control Structures, Repetition Structures. Numbers and Strings Manipulation: Accessing Strings and numbers, String and number manipulations. Lists, Tuples, dictionaries, and date and time: Python Lists, Python Tuples, Python dictionaries, Date and time. Functions and modules: Defining a Function, Arguments, Scope of variables. File Input and Output: Getting keyboard input, File related operations. Exception handling in Python: Exception handling keywords, Exception handling. Graphical User Interfaces: Different related components, Assembling them to make a system in a methodical way. Event handling in Python and introduction to MySQL and databases: Event handing process, Event handling programming. Database Access In Python: Programming In database access, MySQL and databases.

Methods of Evaluation: End of semester examination (60%) Assignments (40%)

Suggested References:

Guttag, J. V. Introduction to Computation and Programming Using Python (Revised and Expanded Edition), MIT Press, 2013

Python programming – tutorials point

CS 2001: Internet Technologies (2L, 1P)

Rationale: This course is designed to start you on a path toward future studies in web development and design, whether you want to train for a new career, advance within your degree programme, or simply learn new skills to stay competitive in today's job market.

Prerequisites: Fundamentals of Database Systems

Intended Learning Outcomes: Upon completion of this course, students will be able to:

- Identify the main concepts in the web development environment
- Design web pages using HTML and CSS
- Use JavaScript to increase the interactivity of the web pages
- Create server scripts using PHP
- Use of SQL and Database Concepts
- Discuss the skills and project-based experience needed for entry into web design and development careers
- Use a variety of strategies and tools to create websites.

Course Content:

Introduction to the Internet and Web - What is the Internet? , What is the World Wide Web?, Web servers and web browsers, IP address and Domain Name System (DNS), URI and URL, Client-server model, HTTP protocol , Standards bodies, The World Wide Web Consortium (W3C) , Web page design with HTML and CSS; HTML - Overview of HTML (V4 and V5), Structure of HTML Document, Structure of HTML Document, Grammar of the HTML, How to save a HTML Page; Displaying Text Data- The Header and the Title, Basic Text Display, Font Sizes and Colors, Font Effects, Paragraphs and Horizontal Rules; Lists- Ordered List, Unordered List, List Attributes (types), Nested Lists, Definition List; Tables- Table Creation, Displaying Table Data and Headings, Attributes of the Table Tag, Cell Split and Cell Merge, thead, tbody, tfoot tags; Displaying Image Data- Displaying Inline Images, Flowing Text around the Image, Attributes of the Image Tag; Specifying Links - Link to other pages (hyperlink/hypertext), Link to the same Page, Link to External Media Data, Link to the external page and target to a specific area, Link to image, Absolute and relative path; Image Map; Using Forms- Textboxes (password), Pull-down Menus, Radio Buttons, Checkboxes, Input Areas, File Field, Send Button and Cancel Button, Frames : Using frames and frame targeting , Inline Frames : iframe, Object Function, script Support, Testing, HTML; CSS: Defining styles using CSS, Styles and HTML, Selectors,child selector (>),adjacent sibling selector (+), general sibling selector; Client-side programming with JavaScript - Client-side programming languages, Uses of JavaScript, Incorporating JavaScript in a HTML document, Basic JavaScript syntax, Data types and variables, Expressions and operators, Control structures, Functions and procedures, Arrays and objects, Document object model (DOM), Event handling, Using JavaScript for form validation ; Server-side programming with PHP-Server side programming languages, PHP fundamentals, Data types and variables, Expressions and operators, Control structures, Functions and procedures, Arrays and objects, Form data handling, Handling sessions, Accessing a MySQL database, Errors and troubleshooting, Web servers and Database Server, application servers and web application servers: e.g. Apache web server; SQL and Database Concepts- Reasons to have Relational Databases, RDBMS terminology, Database Normalization, Relationships, Explore basic commands and functions of SQL, How to use SQL to create tables, How to use SQL for data manipulation

Methods of Evaluation: End of semester examination (60%) Practical Assignments (40%)

Suggested References:

- HTML5 Black Book
- PHP 5.1 for Beginners (Ivan Bayross Sharanam Shah)
- Beginning PHP:Dave

CS 2002: Fundamentals of Software Engineering (3L, 45L)

Rationale: This course provide a general introduction to software engineering. Important concepts such as software processes and agile methods are introduced and essential software development activities from initial software specification through to system evolution are described.

Prerequisites: None

Intended Learning Outcomes: Upon completion of this course, students will be able to:

- Define what software engineering is and why it is important
- Identify suitable software engineering techniques for development of different types of software
- Recognize the ethical and professional issues that are important for software engineers
- Define different software processes/software process models and select suitable software process models for different requirements
- Explain the fundamental process activities of software requirements engineering, software development, testing, and evolution
- Explain why processes should be organized to cope with changes in the software requirements and design
- Explain the rationale for agile software development methods, the agile manifesto, the differences between agile and plan driven development and the issues and problems of scaling agile development methods to the development of large software systems.
- Demonstrate how requirements may be organized in a software requirements document
- Explain why requirements management is necessary and how it supports other requirements engineering activities
- Distinguish different software requirements and to discuss the processes involved in discovering and documenting these requirements.
- Produce system models that may be developed as part of the requirements engineering and system design processes.
- Define concepts of software architecture and architectural design.
- Explain the Object-oriented software design using the UML and illustrate the important implementation concerns
- Define and explain software testing processes
- Explain why software evolution is an important part of software engineering and to describe software evolution processes

Course Content:

Introduction - Professional software development, Software engineering ethics
Software process: Software process models, Process activities, Coping with change, The rational unified process, Agile software development - Agile methods, Plan-driven and agile development, Extreme programming, Agile project management, Scaling agile methods, Requirements engineering - Functional and non-functional requirements, The software requirements document, Requirements specification, Requirements engineering processes, Requirements elicitation and analysis, Requirements validation, Requirements management, System modeling- Context models, Interaction models, Structural models, Behavioral models, Model-driven engineering, Architectural design - Architectural design decisions, Architectural views, Architectural patterns, Application architectures, Design and implementation - Object-oriented design using the UML, Design patterns, Implementation issues, Open source development, Software testing - Development testing, Test-driven development, Release testing, User testing, Software evolution - Evolution processes, Program evolution dynamics, Software maintenance, Legacy system management

Methods of Evaluation: End of semester examination (70%) Assignments (30% - 1 In-class Assignment + 1 Take-home assignment)

Suggested References:

- Sommerville I., " Software Engineering", 9th edition
- Timothy C. Lethbridge and Robert Laganiere, "Object - oriented Software Engineering- Practical Software Development using UML and Java".
- Jeffrey L Whitten & Lonnie D Bentley, "Systems Analysis and Design Methods".