ثانيا: المحتوى العلمي لمقررات قسم علوم الحاسب

CS100: Introduction to Computational Thinking

Computational thinking (CT) is a problem-solving process with the aid of computer; i.e. formulating a problem and expressing its solution in such a way that a computer can effectively carry it out. It includes several characteristics, such as breaking a problem into small and repetitive ordered steps, logically ordering and analyzing data and creating solutions that can be effectively implemented as algorithms running on computer. As such, computational thinking is essential not only to the Computer Science discipline, it can also be used to support problem solving across all disciplines, including math, science, engineering, business, finance and humanities. The aim of this course is hence to take students with no prior experience of thinking in a computational manner to a point where they can derive simple algorithms and code the programs to solve some basic problems in their domain of studies. In addition, the course will include topics to appreciate the internal operations of a processor and raise awareness of the socio-ethical issues arising from the pervasiveness of computing technology. The course also includes learn to design, write, debug, and run programs encoded in the Python language. Develop a working knowledge for how computers operate and how computer programs are executed. Evolve critical thinking and problem-solving skills using an algorithmic approach. Learn about the programmer's role in the software development process. Translate real-world issues into computer-solvable problems.

CS101: Computer Programming I

Introduces the fundamental concepts of procedural programming. Topics include data types, control structures, functions, arrays, files, and the mechanics of running, testing, and debugging. The course also offers an introduction to the historical and social context of computing and an overview of computer science as a discipline. The course also includes Fundamental programming constructs: Syntax and semantics of a higher-level language; variables, types, expressions, and assignment; simple I/O; conditional and iterative control structures; functions and parameter passing; structured decomposition. Algorithms and problem-solving: Problem-solving strategies; the role of algorithms in the problem-solving process; implementation strategies for algorithms; debugging strategies; the concept and properties of algorithms.

CS102: Computer Programming II

Introduces the concepts of object-oriented programming to students with a background in the procedural paradigm. The course begins with a review of control structures and data types with emphasis on structured data types and array processing. It then moves on to introduce the object-oriented programming paradigm, focusing on the definition and use of classes along with the fundamentals of object-oriented design. Other topics include an overview of programming language principles, simple analysis of algorithms, basic searching and sorting techniques, and an introduction to software engineering issues. The course also includes Review of control structures, functions, and primitive data types. Object-oriented programming: Object-oriented design; encapsulation and information hiding; separation of behavior and implementation;

classes, subclasses, and inheritance; polymorphism; class hierarchies. Fundamental computing algorithms: simple searching and sorting algorithms (linear and binary search, selection and insertion sort). Fundamentals of event-driven programming; Introduction to computer graphics: Using a simple graphics API; Overview of programming languages: History of programming languages; brief survey of programming paradigms

CS103: Discrete Structures

Introduces the foundations of discrete mathematics as they apply to computer science, focusing on providing a solid theoretical foundation for further work. Topics include functions, relations, sets, simple proof techniques, Boolean algebra, propositional logic, digital logic, elementary number theory, and the fundamentals of counting. The course also includes Introduction to logic and proofs: Direct proofs; proof by contradiction; mathematical induction. Fundamental structures: Functions (surjections, injections, inverses, composition); relations (reflexivity, symmetry, transitivity, equivalence relations); sets (Venn diagrams, complements, Cartesian products, power sets); pigeonhole principle; cardinality and countability. Boolean algebra: Boolean values; standard operations on Boolean values; de Morgan's laws. Propositional logic: Logical connectives; truth tables; normal forms (conjunctive and disjunctive); validity. Digital logic: Logic gates, flip-flops, counters; circuit minimization. Elementary number theory: Factorability; properties of primes; greatest common divisors and least common multiples; Euclid's algorithm; modular arithmetic; the Chinese Remainder Theorem. The course also includes Predicate logic: Universal and existential quantification; modus ponens and modus tollens; limitations of predicate logic. Recurrence relations: Basic formulae; elementary solution techniques. Graphs and trees: Fundamental definitions; simple algorithms; traversal strategies; proof techniques; spanning trees; applications. Matrices: Basic properties; applications.

CS200: Data Structures

Specification, representation, and manipulation of basic data structures: linked lists, arrays, stacks, queues, trees, strings, symbol tables, Huffman codes, optimal search trees, pattern matching, priority queues, heaps, hash tables. Storage allocation, garbage collection, compaction, reference counts, Sorting, graphs (graph traversal, directed graphs). List and string processing languages. Analysis of algorithms. Performance evaluation involving worst case, average and expected case, and amortized analysis. Students are required to write programs in several languages such as C++, C#, Java, or Pascal.

CS201: Operating Systems

This course will introduce operating system design and implementation. The course will start with a brief historical perspective of the evolution of operating systems over the last fifty years, and then cover the major components of most operating systems. This will include: Computer system structures, Operating system structures, Process and Process management: process synchronization and mutual exclusion; two- process solution and Dekker's algorithm, semaphores (producer- consumer, readers-writer, dining philosophers, etc.), Interprocess communication, Process synchronization, Deadlocks, thread management, CPU scheduling: multiprogramming and time-sharing, scheduling approaches (SJF, FIFO, round robin, etc.),

Memory hierarchy and management: with and without swapping, virtual memory-paging and segmentation, page replacement algorithms, implementation., Virtual memory, Secondary storage management, I/O device management, File system: interface and implementation, FS services, disk space management, directory and data structure, Protection and security, and Case studies: Linux and Windows.

CS202: Analysis and Design of Algorithms

An introduction to the design and analysis of algorithms. The course covers design techniques, such as dynamic programming and greedy methods, as well as fundamentals of analyzing algorithms for correctness and time and space bounds. Topics include advanced sorting and searching methods, graph algorithms and geometric algorithms, notion of an algorithm: big-O, small-O, theta and omega notations. Space and time complexities of an algorithm. Fundamental design paradigms: divide and conquer, branch and bound, backtracking, dynamic programming greedy methods, simulation. Theory of NP-completeness, notion of an intractable problem. Measures of approximation: ratio bound and relative error. Polynomial time approximation scheme. Illustrative examples: graph theory, computational geometry, optimization, numerical analysis and data processing. Other areas vary from year to year, and may include matrix manipulations, string and pattern matching, set algorithms, polynomial computations, and the fast Fourier transform.

CS203: Artificial Intelligence

This is an introductory AI course. Topics will include Artificial and human intelligence, Overview of Artificial Intelligence, Basic Problem-Solving Strategies, Heuristic Search, Problem Reduction and AND/OR Graphs, domains of AI- symbolic processing: semantic nets, modeling model-based reasoning, frames. Knowledge Representation, Representing Knowledge with If-Then Rules. Inference Engines, Inference techniques: implication, forward and backward chaining, inference nets, predicate logic, quantifiers, tautology, resolution, and unification. Rule based systems: inference engine, production systems, problem solving, planning, decomposition, and basic search techniques. AI languages: symbolic and coupled processing prolog: objects and relations, compound goals, backtracking, search mechanism, dynamic databases, lisp, program structure and operations, functions, unification, memory models. Fields of AI: heuristics and game plying, automated reasoning, problem solving, computational linguistics and natural language processing, computer vision, intelligent agents, robotics AI based computer systems: sequential and parallel inference machines, relation between AI and artificial neural nets, fuzzy systems.

CS300: Human Computer Interaction Design

Introduction to Human-Computer Interaction, or how computers communicate with people. Methodology for designing and testing user interfaces, interaction styles (command line, menus, graphical user interfaces, virtual reality), interaction techniques (including use of voice, gesture, and eye movement), design guidelines, and user interface management system software. Comprehensive coverage of computer human interaction(CHI) importance, design, theories, and

future direction; modeling compute interfaces, empirical techniques for task analysis and interface design of interaction, The scope of HCI: Different theories and disciplines that contribute to HCI, HCI Analysis: User analysis, task analysis, environment and domain analysis, Human Cognitive Architecture: Perception, memory, problem solving, Dialogue design: Input, output devices and ergonomics; embedded systems; web usability; interfaces for mobile devices; future systems, CSCW, Influences on Design: Guidelines and standards in HCI; conceptual design, Prototyping in HCI: vertical, horizontal, full, throw-away prototypes, and Empirical evaluation: qualitative and quantitative methods of collecting data from users; the Usability Engineering approach; research topics in evaluation techniques. Students will design a small user interface, program a prototype, and then test the result for usability.

CS301: Systems Programming

Low-level programming; review of addresses, pointers, memory layout, and data representation; text, data, and bss segments; debugging and hex dumps; concurrent execution with threads and processes; address spaces; file names; descriptors and file pointers; inheritance; system calls and library functions; standard I/O and string libraries; simplified socket programming; building tools to help programmers; make and make files; shell scripts and quoting; Unix tools including sed, echo, test, and find; scripting languages such as awk; version control; object and executable files (.o and a.out); symbol tables; pointers to functions; hierarchical directories; and DNS hierarchy; programming embedded systems.

CS302: Formal Languages and Automata

Alphabets and languages. Finite representation of language. Deterministic and non-deterministic finite automata and their applications. Equivalence considerations. Regular expressions. Context-free languages. Context-free grammars. Regular languages, pushdown automata. Properties of context-free languages. Determinism and parsing top-down parsing, and bottom-up parsing. Turing machines: Computing with Turing machines, combining Turing machines, and nondeterministic Turing machines.

CS303: Machine Learning

Machine Learning is concerned with computer programs that automatically improve their performance through experience. Machine Learning methods have been applied to problems such as learning to drive an autonomous vehicle, learning to recognize human speech, and learning strategies for game playing. This course covers the primary approaches to machine learning from a variety of fields, including inductive inference of decision trees, neural network learning, statistical learning methods, genetic algorithms, bayesian methods, explanation-based learning, and reinforcement learning

CS304: Software Testing and Maintenance

Techniques and methods for developing and extending correct, stable, maintainable and efficient software. Testing methodologies and their practical application in software development. Different aspects of testing: Black box testing where testing is done without knowledge of how

the program is written; white box testing where the developer tries to guarantee that every statement, execution path and method is executed during the testing and finally unit testing which is a practical design methodology where test cases are developed as each function or method is written. Software developing aids and methods such as code-inspection. Code and memory profiling as a support for program optimizing.

CS400: High Performance Computing

This course is an introductory course on high-performance computing. High-performance computing refers to a specialized use and programming of (parallel) supercomputers, computer clusters, and everything from software to hardware to speed up computations. The CPU clock speed of desktop and commodity processors has reached a maximum range, due to physical limitations. As a result, more advanced (and often creative) use of software and parallel hardware is required to further speed up processing. In this course you will learn how to write faster code that is highly optimized for modern multi-core processors and clusters, using modern software development tools, performance profilers, specialized algorithms, parallelization strategies, and advanced parallel programming constructs in OpenMP and MPI.

CS401: Compiler Design

The Structure of a Compiler course, Lexical Analyzer, LEX, Design of Lex, Top down Parsing, LL(1) Parsers, Bottom up Parsing, YACC, LR parsers, Syntax Directed Translation, Types and Type Checking, Run-Time Storage Administration and Symbol Table Management, Intermediate Code and Code Generation, Data-Flow Analysis, Code Optimizations, Architecture and recent development on compilers

CS402: Distributed and Concurrent Algorithms

Goals of the course: To present fundamental algorithms and impossibility results from the concurrent programming literature, and to cover techniques for formally specifying and verifying concurrent systems. Both message-passing and shared-memory models of concurrency will be considered. At the end of the course, students will have a general knowledge of the concurrent programming literature and will be able to develop new concurrent algorithms and verify their correctness. Perhaps the most important skill to be developed is the ability to intuitively “see” how or why a concurrent program works (a skill most students probably take for granted when it comes to sequential programs). In other words, this class will teach you how to \think" concurrently.

CS403: Neural networks and deep learning

Over the past few years, neural networks have enjoyed a major resurgence in machine learning, and today yield state-of-the-art results in various fields. This course introduces deep neural network models, and surveys some the applications of these models in areas where they have been particularly successful. The course covers feedforward networks, convolutional networks, recurrent and recursive networks, as well general topics such as input encoding and training

techniques. The course also provides acquaintance with some of the software libraries available for building and training deep neural networks.

CS404: Bioinformatics

Introduces bioinformatics concepts and practice. Topics include biological databases, sequence alignment, gene and protein structure prediction, molecular phylogenetics, genomics and proteomics. Students will gain practical experience with bioinformatics tools and develop basic skills in the collection and presentation of bioinformatics data, as well as the rudiments of programming in a scripting language.

CS405: Fuzzy Logic and Intelligent Systems

Fuzzy Set and Fuzzy Logic: motivation, possibilistic interpretation, basic concepts, set operations, fuzzy relations, and fuzzy inferences. Fuzzy Logic Applications: approximate reasoning, fuzzy arithmetic, linguistic models, decision theory, classification, and fuzzy controllers (development, tuning, compilation, deployment). Computational Intelligence (CI): hybrid systems based on fuzzy, neural and evolutionary computation. Case studies of real world industrial and financial applications.

CS406: Software Design and Architecture

This course is concerned with the principles and concepts of engineering of large software systems and programs. Software architecture is an abstraction of system details that helps in managing the inherent complexity of software systems development. Software architecture provides opportunities for early evaluation of user needs, analysis of requirements and design, and prediction of system properties. Architectural styles, views, notations, and description languages provide systematic frameworks for engineering decisions and design practices. The focus of the course is on advanced topics related to software architecture practices, technologies, and artifacts. Students participate in individual or group projects related to developing architectural representations of software systems.

CS407 Natural Language Processing

Foundations of the natural language processing, language data in corpora, levels of description: phonetics and phonology, morphology, syntax, semantics and pragmatics. Traditional vs. formal grammars: representation of morphological and syntactic structures, meaning representation. context-free grammars and their context-sensitive extensions, DCG (Definite Clause Grammars), CKY algorithm (Cocke-Kasami-Younger), chart-parsing. Problem of ambiguity. Electronic dictionaries: representation of lexical knowledge. Types of the machine-readable dictionaries. Semantic representation of sentence meaning. The Compositionality Principle, composition of meaning. Semantic classification: valence frames, predicates, ontologies, transparent intentional logic (TIL) and its application to semantic analysis of sentences. Pragmatics: semantic and pragmatic nature of noun groups, discourse structure, deictic expressions, verbal and non-verbal contexts. Natural language understanding: semantic representation, inference and knowledge representations.

CS408: Soft Computing

Evolutionary computation (EC), neuro-computation (NC) and fuzzy logic (FL), are considered as three major components of the so-called soft computing. The main idea of soft computation is to make decisions based on rough (incomplete, noisy, uncertain) data. The computing technology which make decisions based on clean, clear and complete data is often called hard computing, although researchers in this field are not hard at all (they are the most intelligent and flexible people in the world). The human brain is a computing machine consisting of two parts. The left part is good at hard computing (logical thinking), and the right part is good at soft computing (heuristic thinking). During the last half century, we developed a lot of computers for assisting the left part of the brain. In this century, we will put more energy to make computers to assist the right part of the brain.

CS409: Introduction to Cryptography

Cryptography provides important tools for ensuring the privacy, authenticity, and integrity of the increasingly sensitive information involved in modern digital systems. Nowadays, core cryptographic tools, including encryption, message authentication codes, digital signature, key agreement protocols, etc., are used behind millions of daily on-line transactions. In this course, we will unveil some of the "magic" of cryptography. Modern Cryptography uses mathematical language to precisely pin down elusive security goals, design primitives and protocols to achieve these goals, and validate the security of designed primitives and protocols using mathematical proofs based on clearly stated hardness assumptions. Therefore, to learn cryptography, it is essential to understand its mathematical underpinning. In this class, we will see the inner working of cryptography for several core cryptographic tools, from encryption, to message authentication codes, to hash functions, to digital signatures, etc.

CS410: Theory of Computation

An introduction to the theoretical foundations of computing, including abstract models of computing machines, the grammars those machines recognize, and the corresponding classes of languages. Topics include: Church's thesis; Grammars, the M-recursive functions, and Turing computability of the M-recursive functions, The incompatibility: The halting problem, Turing innumerability, Turing acceptability, and Turing decidability, unsolvable problems about Turing machines and M-recursive functions, Computational complexity: Time-bounded Turing machines, Rate of growth of functions, NP- Completeness, The complexity hierarchy, The prepositional calculus: Syntax, Truth-assignment, Validity and satisfy, and Equivalence and normal forms compactness.

CS411: Programming Language Design

This course is an introduction to the principles which underlie the definition and implementation of programming languages. Study of modern programming language paradigms (procedural, functional, logic, object oriented). Introduction to the design and implementation of programming languages including syntax, semantics, data types and structures, control structures, and run-time environments.

CS4112: Intelligent Agents

This course gives a broad introduction to the new and rapidly expanding field of agent-based computing. It introduces the key concepts and models of the field, dealing both with the individual agents and with their interactions. Emphasis is placed on automated negotiation, cooperation and on-line auctions, and students are required to program a trading agent in Java which will compete in a class tournament within a simulated trading environment.

CS413: Computer Systems Performance

It introduces the main concepts and techniques needed to plan the capacity of computer systems, predict their future performance under different configurations, and design new applications that meet performance requirements. The course is mainly based on the use of analytic queuing network models of computer systems. These techniques are applied to study the performance of centralized, distributed, parallel, client/server systems, Web server and e-commerce site performance. The course also discusses performance measuring tools for operating systems such as Unix and Windows.

CS425: Selected Topics in Computer Science I

Selected Topics provides an opportunity to study a topic which is not included in the existing curriculum. This course examines one or more selected current issues in the area of Computer Science. Topics chosen for study will be by arrangement with the department.

CS426: Selected Topics in Computer Science II

Selected Topics provides an opportunity to study a topic which is not included in the existing curriculum. This course examines one or more selected current issues in the area of Computer Science. Topics chosen for study will be by arrangement with the department.

CS430/CS431: Project

This course will continue for two semesters. In the first semester, a group of students will select one of the projects proposed by the department and analyze the underlying problem. In the second semester, the design and implementation of the project will be conducted. The student will deliver oral presentations, progress reports, and a final report.