CE/CZ 1003 – Introduction to Computational Thinking

Course Code	CE/CZ 1003					
Course Title	Introduction to Computational Thinking					
Pre-requisites	NIL					
No of AUs	3					
Contact Hours	Lectures	0	TEL (Online Videos)	13	Example Class (Seminars/Hands-on Exercises)	26

Course Aims

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 a number of 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.

Intended Learning Outcomes (ILO)

Upon the successful completion of this course, you shall be able to:

1. Describe the internal operation of a basic processor, how a program is executed by a computer and computing trends.
2. Code basic programs based on the programming language used in the course
3. Formulate a problem and express its solution in such a way that a computer can effectively carry it out. (i.e. equip you with CT skills)
4. Apply the CT concepts on case studies/problem-based scenarios through hands-on practice of the CT processes.

Course Contents

	Topics	LAMS (Hours)	Example Class (2-hr session)
0	**Course Overview and Concepts of Computational Thinking** Solving complex problem using computer **- enables the student to work out exactly what to tell the computer to do.**	0.5	RPi Familiarization Scratch based graphical programming (1 week)
1	**Overview of Programming Languages** Graphic programming, high level programming languages (Python, C, Java, R, Matlab)	0.5	
2	**Basic internal operation of computer** Basic computer organization (Processor, Memory, I/O) and how a computer execute a program (Machine instructions)	1	Python on RPi (1 week)
3	**Basic program structure: control constructs and data types** Concepts of data types, variables; Pseude code and flowcharts; Sequences, Selection (if/else), iteration (for/while loop);	3	Python Programming Exercises/Quiz (3 weeks)
4	**CT concept - Abstraction** Problem formulation - reducing something to a very simple set of characteristics to only focusing on the most relevant to the problem. Concept of functions/libraries and data structure	2	Library calls & function (2 weeks)
5	**CT concept - Decomposition** Break a complex problem into smaller and more manageable parts/steps, such that each of these smaller problems can then be looked at individually	1	Project – Flow Chart Design (1 week)
6	**CT concept – Pattern recognition** Looking for similarities among and within problems, which also enable re-use knowledge of previous similar problems	1	Project – Coding (1 week)
7	**CT concept – Algorithm** Reformulating the problem into series of ordered steps through Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources. (Some common/useful examples: Sorting and searching)	2	Project – Coding/Quiz (2 weeks)
8	**Limit of computing** Analysis of Algorithm Complexity to determine how much resources (space and time) are needed to execute an Algorithm in order to achieve code optimization.	1	Project – Demo and assessment (1 week)
9	**Computing Trends** Cloud, Edge and Fog computing, Quantum Computers	0.5	Project – Demo and

10	**Social-Ethical Issues and Ramifications of Computing** Fairness, Privacy, Sharing, Hacking, software Piracy, Data Protection, Cyberbullying and trolling, Fake news, digital divides, IP/Copyright	0.5	assessment (1 Week)
	Check for Hours	13	26

Assessment (includes both continuous and summative assessment)

a) TEL participations and TEL MCQs: 20%
b) Online MCQs based quizzes: 30%
c) Hands-on exercises assessment (MCQ based quizzes): 20%
d) Mini Project assessment: 30%

CE/CZ 1005 – Digital Logic

Course Code	CE/CZ 1005							
Course Title	Digital Logic							
Pre-requisites	NIL							
No of AUs	3							
Contact Hours	Lectures	0	TEL	26	Tutorials	13	Lab	10

Course Aims

This course aims to develop your ability to analyse and design digital circuits. The design and behaviour of modern digital equipment (e.g. mobile phones and personal computers) are based on functions of binary variables. This course provides you the knowledge to analyse the theoretical characteristics and the skills required for practical technology-driven implementations of digital circuits.

This course provides an introductory perspective of digital circuits design. It is relevant for anyone pursuing a career in the ICT industry – including those in digital systems, computer architecture and microprocessor based systems design.

Intended Learning Outcomes (ILO)

Upon the successful completion of this course, you shall be able to:

1. Represent integers in binary.
2. Recognise and use elementary logic operators.
3. Analyse a given Boolean expression and optimize it for efficient implementation.
4. Design and implement simple combinational and sequential circuits.
5. Use Verilog to design hardware systems based on the above principles.

Course Content

	Topics	Lectures (Hours)	Tutorials (Hours)
1	**Binary integers and arithmetic** Unsigned binary integers. Hexadecimal and Binary-Coded Decimal. Signed formats including sign-magnitude, offset, and 2's complement. Gray codes and Hamming distances. Binary addition, subtraction, and overview of multiplication.	3	1
2	**Boolean Variables and Logic** Boolean variables; truth tables for AND, OR and exclusive-OR. Operators; algebra and theorems, including associativity, commutativity distributivity and consensus; derivation of DeMorgan's Duals; circuit representations/symbols and multi-input (>2) gates.	3	2
3	**Combinational circuits** Sum-of-Products vs Product-of-Sums; optimisations with respect to gate count, type, and delays; Karnaugh maps and simplification. Application and gate-level design of data multiplexers, demultiplexers, decoders and parity generators. Cascaded designs. Half-adder; full-adder; multi-bit adders, comparators.	4	2
4	**Implementation technologies** History and evolution. Comparison of Transistor-Transistor Logic and Complementary Metal Oxide Semiconductor (CMOS). Consideration of voltages, current, power dissipation, propagation delay, clock rates, noise immunity. Moore's Law. Active high vs active low signals. Buffers and Schmitt triggers.	2	0
5	**Digital design using hardware description languages** Introduction to Verilog (and VHDL) hardware description languages (HDLs). Structural Verilog, module instantiation and design hierarchy. Combinational and Sequential circuits in Verilog. Finite State Machines in Verilog. Overview of Synthesis and Simulation.	5	3

6	**Sequential circuits** Combinatorial circuits with feedback; Set-Reset latch; enabled latches; master-slave positive-edge-triggered D-Type flip-flop; propagation delays, set-up and hold-times, meta-stability, race conditions; asynchronous clear and set.	3	2
7	**Sequential circuits to building blocks** Multi-bit registers, binary counters and dividers; synchronous and asynchronous loading, shift registers, parallel-to-serial and serial-to-parallel conversion; addressable memories.	3	1
8	**Finite state machines** State diagrams; state assignments; state transition logic; Mealy and Moore machines; dead ends and contradictions.	3	2
	Check for Hours	=26	=13

Assessment (includes both continuous and summative assessment)

a) Final Examination: 60%
b) Quizzes: 22%
c) Lab assessments: 10%
d) Online tasks: 8%

CE/CZ 1006 – Computer Organisation and Architecture

Course Code	CE/CZ 1006							
Course Title	Computer Organisation and Architecture							
Pre-requisites	NIL							
No of AUs	3							
Contact Hours	Lectures	26	TEL	-	Tutorials	13	Laboratories:	8

Course Aims

This core engineering course aims to develop your understanding of the basic architectures of a computer system along with the organization and operation of common peripherals and memory sub-systems around such computing systems. It also reveals how such architectures influence the design and implementation of low-level instructions and high level programming constructs.

Understanding of low level architectural aspect is important to later appreciation of more advanced topics in computer system and architecture discussed in CE2007, CE3001, CZ3001 and CE3003, which this course is a pre-requisite to.

Intended Learning Outcomes (ILO)

By the end of this course, the student would be able to:

1. Describe and analyse the basic factors influencing computer organization and performance.
2. Describe the basic operation of the CPU and how programs are executed on a computer
3. Describe and explain how high-level programming constructs are implemented with low-level instructions
4. Describe, analyse and evaluate the characteristics of different memory subsystems
5. Describe and analyse basic input/output techniques
6. Describe and evaluate the different number representations and basic arithmetic operations.

Course Content

	Topics	Lectures (Hours)	Tutorials (Hours)
1	**Computer Hardware Decomposition** Interconnection of processor, memory, data storage, power supplies and peripherals. Input devices such as touch screens, mice, and keyboards and output devices.	1	0
2	**Data Representation, Memory Allocation and Access** Unsigned Integers. Addressing bytes, words, arrays and structures. Little Endian vs Big Endian. Pointers and pointer arithmetic. Character encoding (DEC 6-bit vs ASCII vs Unicode)	1	1
3	**Central Processing Unit** Concept of a stored program. Allocation of memory to code and data segments in Von Neumann architecture and dual memory spaces in Harvard architecture with concurrent access. Programmer's functional models of processors. Status register and flags. Control unit. Fetch-decode-execute cycle.	2	1
4	**Assembly Programming and Instruction Set Architecture** Assembly-level programming. Addressing modes. Use of Program Counter for instruction and data access. Basic arithmetic and logical operations. Program control instructions. Supervisor vs User modes. Privileged instructions. Instruction set architectures (ISA) and paradigms. Classifications based on accumulators, registers, load-store and memory-to-memory transfers and operations. Differentiating characteristics of 8-bit to 64-bit ISAs. Fixed-length vs variable-length instructions. Mnemonics, instruction encoding and orthogonality.	5	3
5	**High-level Software to Low-level Instructions** Mapping of high-level language constructs for control flow and data structures to machine-level operations. Including 'if', 'for' and 'while' with pre- and post- test conditions. Using examples of code produced by a 'C' compiler. Mechanisms for subroutine/function linkage, including return address, parameter/result passing and local data. Use of Stack Pointer, Stack Frame and Link Register. Position-independent functions. Introduction to basic compiler optimization techniques.	4	2

6	**Computer Memory** Memory Hierarchy (System, Virtual and Storage). Volatile and non-volatile Random Access Memory. Static RAM versus Dynamic RAM. Principles of Cache Memory. Principles of Virtual Memory (Paging versus Segmentation). Principles of accessing semiconductor memories with unidirectional addressing and control buses, and bi-directional data bus. Latency, bandwidth and interleaving. Operating principles of magnetic rotating disc drives versus solid-state drives using semiconductor NAND flash drives. Logical versus physical memory access. Remapping, reliability and redundancy. Sequential/paged NAND flash memory.	4	2
7	**Data Transfer and Input/Output (I/O) Techniques** Data transfer principles, I/O interfaces, Evolution of data interfacing. Asynchronous versus synchronous data transfer. Sources of interference. Serial versus parallel data transfer. Concept of handshaking and buffering. Programmed I/O. Interrupts and interrupt service routines. Direct Memory Access (DMA).	4	2
8	**Computer Arithmetic** Positional number systems. Binary to/from decimal conversion. Signed integer representation. Fixed-point arithmetic. Carry vs. Overflow. IEEE 754-2008 standard. Special cases and exceptions. Range and precision. Floating-point arithmetic.	3	2
9	**Measuring system performance** Computer performance/speedup/energy equations. MIPS/FLOPS and limitations. Benchmarks and relevance or irrelevance to applications. Software tools for code profiling.	1	0
10	**Towards higher speed** Pipelines, co-processors, superscalar and multicores. Single Instruction Multiple Data (SIMD) vs. Multiple Instruction Multiple Data (MIMD). Homogeneous and heterogeneous multiprocessors.	1	0
	Check for Hours	=26	=13

Assessment

a) Final Examination: 60%
b) Quiz: 40%

CE/CZ 1007 – Data Structures

Course Code	CE/CZ 1007							
Course Title	Data Structures							
Pre-requisites	CE/CZ 1003: Introduction to Computational Thinking							
No of AUs	3							
Contact Hours	Lectures	13	TEL	0	Example classes	24		

Course Aims

This core programming course aims to develop your understanding in data structures such as linked lists, stacks, queues and trees that are important for building efficient programs in C programming language, and are essential for future programming and software engineering courses.

Intended Learning Outcomes (ILO)

By the end of this course, the student would be able to:

1. Design and implement simple C programs using basic programming constructs.
2. Design and implement pointers, arrays, character strings and structures in C.
3. Design and implement recursive functions in C.
4. Design and implement dynamic memory allocation in C.
5. Design and implement linked lists, stacks and queues in C.
6. Design and implement tree structures in C.

Course Content

	Topics	Lectures (Hours)	Example Classes (Hours)
1	**Basic Programming Constructs in C Language** C program structure. Syntax and semantics. Intrinsic data types, declarations, operators, assignments, control flow, and simple input/output. Pre-processing. Functions. Return values, arguments and parameter passing. Scopes of variables. Concept of side effects.	1	2
2	**Built-in Data Structures** Pointers, pointer operations and pass by reference. One-dimensional and multi-dimensional arrays, and pointers and arrays. Character strings and arrays of strings. Structures, arrays of structures and type definitions.	5	10
3	**Recursion** Basic recursion. Problem solving with recursion. The Towers of Hanoi. Recursive versus iterative functions.	1	2
4	**Memory Management** Static vs dynamic memory allocation. Overview of node-based data structures.	1	2
5	**Linked Lists** Linked list structures and applications.	1	2
6	**Stacks and Queues** Stack and queue structures and applications. Implementation using linked lists.	1	2
7	**Tree Structures** Overview of hierarchical/non-linear data structures. Tree structures and applications. Binary vs general trees. Tree traversal: pre-order, in-order, post-order. Binary search trees. AVL tree balancing. Application-specific trees: parse trees.	3	4
	Check for Hours	=13	=24

Assessment (includes both continuous and summative assessment)

a) Assignment: 20%
b) Test: 80%

CE/CZ 1011 – Engineering Mathematics I

Course Code	CE/CZ 1011					
Course Title	Engineering Mathematics I					
Pre-requisites	NIL					
No of AUs	3					
Contact Hours	Lectures and TEL modules	26	Tutorials	13	Example Class	8

Course Aims

This course aims to develop your foundation in engineering mathematics, particularly in linear algebra and probability and statistics.

This course provides the basic engineering mathematics foundation that is necessary for anyone pursuing a computer science and engineering degree course.

Intended Learning Outcomes (ILO)

This course introduces linear algebra and probability & statistics at a foundation level. Upon the successful completion of this course, you shall be able to:

1. Interpret matrices, determinants, vectors and the various matrix and vector operations.
2. Apply linear algebra to solve practical problems in engineering and science.
3. Present data using appropriate graphs and charts.
4. Apply probability and statistics to understand random phenomenon and make appropriate inferences.
5. Apply numerical methods for data assessment and analysis.

Course Contents

	Topics	Lectures/TEL (Hours)	Tutorials (Hours)
1	**Complex Numbers** Argand diagram, rectangular and polar representations, operations on complex numbers, Euler's formula, De Moivre's theorem, nth root of complex numbers.	3	1
2	**Vectors** Definition of vectors, vector operations, lines and planes in space, vector form of linear systems, vector spaces, geometric interpretations of linear systems and their solutions.	3	2
3	**Matrices** Definition of matrices, matrix operations, scalar products, transpose of a matrix, inverse of a matrix.	3	1
4	**Systems of Linear Equations** Matrix form of linear systems, elementary row operations, Homogeneous and non-homogeneous systems, Gauss-Jordan eliminations, Cramer's rule, applications of linear algebra.	4	2
5	**Descriptive statistics** Types of data, presenting data, measures of central tendency, measures of variation.	2	1
6	**Probability theory** Sample space, events, mutually exclusive and independent events, probability rules, conditional probability, Bayes' theorem.	3	2
7	**Probability and sampling distributions** Random variables, expectation and variance of random variables; discrete probability distributions: Bernoulli, Binomial, geometric and Poisson distributions; continuous probability density functions: uniform and normal distributions; sampling distributions, Central Limit Theorem.	3	2
8	**Inferential statistics** Unbiased point estimates, interval estimates, hypothesis testing of means, z and t tests, means and proportions, Type I, II errors.	2	1
9	**Experimental and Numerical Methods** Linear Regression. Roots of Equations Newton's method, and Bisection method. Software tools – e.g. Scilab, Matlab.	3	1
	Check for Hours	=26	=13

Assessment

a) Final Written Examination: 60%
b) Quizzes: 32%
c) Example class (lab) exercise: 8%

CE/CZ 1012 – Engineering Mathematics II

Course Code	CE/CZ 1012							
Course Title	Engineering Mathematics II							
Pre-requisites	NIL							
No of AUs	3							
Contact Hours	Lectures	20	TEL	6	Tutorials	10	Example class	8

Course Aims

This course aims to provide the appropriate mathematical background pertaining to basic calculus, ordinary differential equations, Fourier transform, numerical differentiation and integration methods. It is a pre-requisite for other courses, including CE2001 and CZ2001, CE2004, and CE3005 and this mathematical knowledge is essential for solving many computer science and computer engineering problems.

Intended Learning Outcomes (ILO)

Upon the successful completion of this course, you shall be able to:

1. Apply analytical and theoretical skills to solve mathematical problems.
2. Apply calculus to solve practical problems in engineering and science.
3. Analyze convergence property of Sequences and Series.
4. Apply Fourier methods to analyse signals.

Course Content

	Topics	Lectures (Hours)	Tutorials and/or Example Classes (Hours)
1	**Precalculus:** Linear functions, polynomial functions, power functions, logarithmic functions, trigonometric functions, floor and ceiling functions, composition of functions, complex numbers	1	2
2	**Limits and Continuity**: Concept of limits, one-sided limits and limits at infinity, methods to find limits, Sandwich theorem, definition of continuity and continuity test.	2	2
3	**Differentiation:** Definition of derivatives, differentiation rules, implicit differentiation, applications of derivatives, L'Hopital's rule	3	2
4	**Integration:** Indefinite integrals, definite integrals, Riemann sums, Fundamental Theorems of Calculus, techniques of integration, applications of integrals.	4	2
5	**Ordinary Differential Equations (ODE):** Classification of differential equations, order, linearity. Solutions of first order linear ODE and second order linear ODE with constant coefficients.	3	1
6	**Sequences and Series:** Sequences, series, convergence and divergence, absolute and conditional convergence, tests for convergence, power series.	2	2
7	**Function approximation:** Taylor's series and Maclaurin series, interpolation, extrapolation, and curve fitting (regression)	4	2
8	**Numerical differentiation and integration:** Simpson's Rule, explicit and implicit methods	2	2
9	**Fourier Series:** Signal representation in an orthogonal basis, trigonometric Fourier series, complex Fourier series.	2	2

10	**Fourier Transform:** Fourier integral, Fourier transform theorems, inverse Fourier transform.	3	1
		=26	=18

Assessment (includes both continuous and summative assessment)

a) Final Examination: 60%
b) Quiz: 20%
c) Take home assignment: 20%

CE/CZ 1013 – Physics for Computing

Course Code	CE/CZ 1013							
Course Title	Physics for Computing							
Pre-requisites	NIL							
No of AUs	2							
Contact Hours	Lectures	26	TEL	0	Tutorials	0	Lab	8

Course Aims

This course aims to show the fundamental role of physics in building up computing systems and computer applications. You will be exposed to various selected physics topics (Optics, Electrics, Electronics, and Magnetism), with which, many useful physics-computing systems have been developed and changed our daily life. This course serves you as an introductory general education course to encourage you for interdisciplinary thinking and exploration.

Intended Learning Outcomes (ILO)

This course introduces physics used in computing systems at an elementary level. Upon the successful completion of this course, you shall be able to:

1. Explain basic knowledge (definitions, principles and techniques) of all the selected topics;
2. Identify the contribution of physics in some computing systems;
3. Discuss the interdisciplinary nature of many innovative systems;
4. Reflect the importance and the recent progress of Physics, to prepare to propose novel physics-computing systems in the future.

Course Content

	Topics	Lectures (Hours)	Labs (Hours)
1	**Introduction: what is physics and why physics?** Background of the course and physics in general	2	0
2	**Optics and Applications** Light as waves and holography; light as rays and 2D/3D imaging; LiDAR and self-driving cars;	6	2
3	**NFC/Electricity and Magnetism** Fundamentals of electricity; current, voltage, power and resistance; electro-magnetism; electric and magnetic fields; example application.	6	2
4	**EEG/Electrical Potential** Fundamentals of measuring electro-physiological signals; circuits and electrodes for EEG recording; EEG signals and their applications.	6	2
5	**Electronics/Semiconductor** Energy bands in solids; electron, hole carriers; P-N junction; diode and transistor switch; semiconductor integrated circuits fabrication.	6	2
	Check for Hours	=26	=8

Assessment

a) Quizzes/Reports: 80%
b) Labs: 20%