Proposed syllabi for B.Tech Computer Science and Engineering (5th to 8th Semester)

<table>
<tr><td>Course Title</td><td>Entrepreneurship and Management Functions</td><td>Course No</td><td colspan="3">To be filled by the office</td></tr>
<tr><td>Specialization</td><td>HMC</td><td>Structure (IPC)</td><td>3</td><td>0</td><td>3</td></tr>
<tr><td>Offered for</td><td>UG</td><td>Status (Core / Elective)</td><td colspan="3">Core</td></tr>
<tr><td>Prerequisite</td><td>Systems Thinking and Design</td><td>To take effect from</td><td colspan="3"></td></tr>
<tr><td>Course Objectives</td><td colspan="5">The objective of this course is to provide engineering students an exposure to the basic concepts of entrepreneurship and management, with a specific focus on the process of turning an idea into a commercially viable venture.</td></tr>
<tr><td>Course Outcomes</td><td colspan="5">At the end of the course, the students will learn how to
Understand the market & competition
Prepare a business case for the product/idea</td></tr>
<tr><td>Contents of the course</td><td colspan="5">Module 1: Introduction
· Division of labor and creation of value
· Evolution of organizations, industries and sectors, for profit and non-profit
· Role of Entrepreneurs and Managers in value creation
· Principles of Management - Planning, Organizing, Resourcing, Directing (4)

Module 2: Strategy & Planning
· Understanding industry dynamics & competition (Porter's Framework)
· Understanding the industry value chain and firm positioning (6)

Module 3: Organizing
· Typical organizational functions (R&D, Marketing & Sales, HR, Operations)
· Cybernetics of organizational functions (Stafford Beer's viable systems model)
· Types of organization structures (product, functional, matrix, global) (6)

Module 4: Resource Management
· Financial management (Sources of funding, how to read a P&L, balance sheet)
· Human resource management (Interviewing, compensation, motivation)
· Global sourcing and supply chain management (8)

Module 5: Management Information & Decision Making (4)

Module 6: Legal and Regulatory environment (4)</td></tr>
<tr><td>Textbook</td><td colspan="5">1. Peter F Drucker, The Practice of Management, Harper Collins, 2006, ISBN: 978-0060878979
2. Hentry Mintzberg, Managing, Berret-Koehler Publishers, 2009, ISBN: 978-1605098746
3. Michael Porter, On competition: Updated and Expanded Edition, HBS, 2008, ISBN: 978-1422126967
4. Vasanta Desai, Dynamics of Entrepreneurial Development and Management, Himalaya Publishing House, ISBN:9788183184113.</td></tr>
<tr><td>References</td><td colspan="5">1. Walter Isaacson, Steve Jobs, 2011, ISBN:978-1451648539
2. Eric Ries, The Lean Startup, Portfolio Penguin, 2011, ISBN: 978-0307887894
3. Vineet Bajpai, Build from scratch, Jaico books, 2013, ISBN: 9788184952919.</td></tr>
</table>

<table>
<tr><td>Course Title</td><td>Operating Systems</td><td>Course No</td><td colspan="3">To be filled by the office</td></tr>
<tr><td>Specialization</td><td>Computer Engineering</td><td>Structure (IPC)</td><td>3</td><td>0</td><td>3</td></tr>
<tr><td>Offered for</td><td>UG</td><td>Status (Core / Elective)</td><td colspan="3">Core</td></tr>
<tr><td>Prerequisite</td><td>Computer Organization and Design</td><td>To take effect from</td><td colspan="3"></td></tr>
<tr><td>Course Objectives</td><td colspan="5">This first level course focuses on exposing students to the purpose, structure and functions of an operating system. Operating systems abstraction, mechanisms and their implementation support for concurrency (threads) and synchronization, resource management, scheduling strategies, etc. are explored.</td></tr>
<tr><td>Course Outcomes</td><td colspan="5">Students shall have a sound understanding of basic concepts relating to the design and implementation of an operating system. Specifics relating to scheduling, multithreading, synchronization, etc. shall help them understand the structure of the operating system (Linux), at the concept and the source code level.</td></tr>
<tr><td>Contents of the course</td><td colspan="5">Functionalities & Services of an Operating System – System Calls & Types - Process Concept – Process Control Block – Linux System calls for Process creation, Inter Process Communication using Shared memory / Message passing. (10)

Concurrency – Multithreaded programming – benefits, challenges, models, Pthreads library in Linux – thread creation, cancellation, thread specific data, Thread pools, Signal handling , Scheduling – Premptive, Non preemptive algorithms FCFS, SJF, SRT, RR – Thread scheduling – contention scope, pthread support for scheduling. (11)

Synchronization – Race condition – Critical Section Problem, Solution, Mutex Locks and Semaphores – Priority Inversion, Pthreads synchronization - Producer Consumer problem (multi threaded) example Deadlock characterization – Resource graph – Avoidance & Prevention – Safe state – Bankers algorithm – recovery schemes. (10)

Memory management – logical v/s physical address space – Segmentation, Paging, Page table structures , Virtual memory, Page replacement strategies, File Systems – file operations, types, access methods, Directory structure, Mounting file systems. (11)</td></tr>
<tr><td>Textbook</td><td colspan="5">1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, Operating System Concepts, John Wiley, 9th Edn, 2015.</td></tr>
<tr><td>References</td><td colspan="5">1. Andrew S Tanenbaum, Modern Operating Systems, Prentice Hall, 2007.
2. Stallings. W, Operating System: Internals and Design Principles, Prentice Hall, 2009.
3. Gary Nut, Operating Systems: A Modern Perspective, Addison Wesley, 2003.</td></tr>
</table>

Course Title	Computer Networking	Course No	To be filled by the office		
Specialization	Computer Engineering	Structure (IPC)	3	0	3
Offered for	UG	Status (Core / Elective)	Core		
Prerequisite	Computer Organization and Design	To take effect from			
Course Objectives	To introduce the basics of computer networking, error detection and correction techniques, and flow control techniques. Also an exposure to IP addressing and routing and its associated protocols would be given. A highlight of various application layer protocols and its relevance in modern networking world would be discussed.				
Course Outcomes	To be able to design a local area network and analyze the network using performance metrics. To appreciate the importance of subnetting, masking, and nuances involved in setting up a campus network.				
Contents of the course	Evolution of computer networks, creating a small network, Data transfer between nodes, encoding of bits in physical layer, NRZ, Manchester, Differential Manchester, Performance evaluation of a network: propagation delay, transmission delay, RTT, effective bandwidth. (10) Error detection techniques in Data link layer (LRC, CRC, Two dimensional parity check), Hamming Error correcting codes. Data transfer between nodes using stop and wait protocol, sliding window protocol (Go-back-n and selective reject), performance analysis of stop and wait and sliding window protocols. Flow control at data link layer. Introduction to layer-2 devices (switches, bridges) and addressing scheme at Layer-2 (MAC addresses). (10) Creating a small network using Ethernet (IEEE 802.3) Token Ring (IEEE 802.5), Performance evaluation of IEEE 802.3 and 802.5 networks. Introduction to Layer-3 devices, IP addresses, IPv4,IPv6, Error detection at layer-3 using Checksum. IP addressing schemes, subnetting, CIDR (12) Introduction to TCP/IP, IP routing, RIP, OSPF, Circuit and Packet switching, ICMP, Introduction to networking commands: Ping, Traceroute, IPconfig, UDP, congestion control and avoidance. (10) Introduction to DHCP, FTP, HTTP and other application layer protocols. (3)				
Textbook	1. Larry L.Peterson and Bruce S Davie, Computer Networks: A systems Approach, Morgan, 3rd Edn, 2003. 2. William Stallings, Data and Computer Communications, 6th Edn, Pearson, 2000.				
References	1. Andrew S. Tanenbaum, Computer Networks, 4th Edn, 2003.				

Course Title	Compiler Design	Course No *(will be assigned)*				
Specialization	Computer Science & Engineering	Structure (LTPC)	3	0	0	3
Offered for	UG	Status	Core ■		Elective ☐	
Faculty		Type	New ☐		Modification ☐	
Pre-requisite		To take effect from				
Submission date		Date of approval by Senate				
Objectives	The objective of this course is to train students to design various phases of compiler such as Lexical analyzer, syntax analyzer, semantic analyzer, intermediate code generator, code optimizer and code generator. Students are also exposed to design compiler construction tools such as Lexical Analyser generator and parser generator.					
Contents of the course	Introduction to phases of compiler – Grouping of phases – DFA – Lexical analysis – Token Specifications (6) Parser – Context free grammar – Types of parsing – Top down and bottom up – Recursive descent – Predictive – Shift reduce – Operator precedence – LR, SLR and CLR, LALR parsers (10) Intermediate code generation – Languages – Declaration – Assignment statements – Boolean expressions – Multiple selection statements – Back patching and procedure calls (8) Code generator design issues – Target machine – Runtime storage management – Basic blocks – Flow graphs – Next use information – Code generator case study – Directed acyclic graph representation of basic blocks – Peephole optimization technique (8) Introduction to code optimization – Sources – Block optimization – Global data flow analysis –Language issues – Storage optimization & allocation strategies – Parameter Passing (12)					
Textbook	1. Alfred Aho, Ravi Sethi and Jeffrey D Ullman, Compilers Principles, Techniques and Tools, Pearson Education, 2003.					
References	1. Levine J.R, Mason T, Brown D, Lex & Yacc, OReilly Associates, 1992. 2. Allen I. Holub, Compiler Design in C, Prentice Hall, 2003. 3. Kamala Krithivasan and R Rama, Introduction to Formal Languages, Automata Theory and Computation, Pearson Education, 2009.					

Course Title	Compiler Design Practice	Course No *(will be assigned)*				
Specialization	Computer Science & Engineering	Structure (LTPC)	0	0	3	2
Offered for	UG & DD	Status	Core ■		Elective ☐	
Faculty		Type	New ☐		Modification ☐	
Pre-requisite		To take effect from				
Submission date		Date of approval by Senate				
Objectives	The aim is to write a compiler for a small language. Familiarity with compiled codes (assembly language) of RISC and CISC machines, writing a scanner, writing a predictive parser for a small language, a small experiment with scanner (lex/flex) and parser (yacc/byson) generator (such as translation of regular expressions to NFA or the construction of parse tree), writing scanner parse specification for a small language, translation of the language to an intermediate form (e.g. three-address code), generation of target code (in assembly language).					
Contents of the course	Lexical analyzer implementation in C - Lexical analyser implementation using LEX tool - Recursive descent parser implementation in C for an expression grammar - YACC and LEX based implementation for an expressions grammar - YACC implementation of a calculator that takes an expression with digits, + and * and computes and prints its value - Front end implementation of a compiler that generates the three address code for a simple language- Back end implementation of a compiler which takes the three address code (output of previous exercise) and results in assembly language instructions - Implementation of peephole optimization in C.					
Textbooks	1. Alfred Aho, Ravi Sethi and Jeffrey D Ullman, Compilers Principles, Techniques and Tools, Pearson Education, 2003. 2. Andrew W. Appel, Modern Compiler Implementation in C/Java, Cambridge University Press. 3. Allen I. Holob, Compiler Design in C, Prentice-Hall.					
References	1. Keith D. Cooper and Linda Torczon, Engineering a Compiler, Elsevier. 2. Steven S. Muchnik, Advanced Compiler Design and Implementation, Elsevier.					

Course Title	Formal Languages & Automata Theory	Course No *(will be assigned)*				
Specialization	Computer Science & Engineering	Structure (LTPC)	3	0	0	3
Offered for	UG& DD	Status	Core ■		Elective ☐	
Faculty		Type	New ☐		Modification ☐	
Pre-requisite	Discrete Structures for Computing	To take effect from				
Submission date		Date of approval by Senate				
Objectives	Course should provide a formal connection between algorithmic problem solving and the theory of languages and automata and develop them into a mathematical (and less magical) view towards algorithmic design and in general computation itself. The course should in addition clarify the practical view towards the applications of these ideas in the engineering part of CS.					
Contents of the course	Finite Automata & Regular Languages : Languages vs Problems. Finite State Automata, Regular Languages. Closure properties, Limitations, Pumping Lemma, Myhill-Nerode relations, Quotient Construction. Minimization Algorithm. (13) Non-determinism & Regular Expressions : Notion of non-determinism. Acceptance condition. Subset construction. Pattern matching and regular expressions. Regular Expressions and Regular languages. More closure properties of regular languages. (8) Grammars & Context-free Languages(CFLs) : Grammars and Chomsky Hierarchy, CFLs, Regular Grammars, Chomsky Normal Form, Pumping Lemma for CFLs, Inherent Ambiguity of Context-Free Languages, Cock-Younger-Kasami Algorithm, Applications to Parsing. Pushdown Automata(PDA), PDA vs CFLs. Deterministic CFLs. (14) Turing Machines & Computability : Introduction to Turing Machines, Configurations, Halting vs Looping. Multi-tape Turing machines. Recursive and Recursively enumerable languages. Undecidability of Halting Problem. Reductions. Introduction to Theory of NP-completeness. (9)					
Textbook	1. Introduction to Automata Theory, Languages and Computation, Hopcroft, Motwani, and Ullman, Pearson Publishers, Third Edition, 2006. 2. Kamala Krithivasan and R Rama, Introduction to Formal Languages, Automata Theory and Computation, Pearson Education, 2009.					
References	1. Automata and Computability, Dexter C. Kozen, Springer Publishers, 2007. 2. Elements of the Theory of Computation, H. R. Lewis and C.H. Papadimitriou, Prentice Hall Publishers, 1981 3. Introduction to Languages and the Theory of Computation, John. C. Martin, Tata McGraw-Hill, 2003.					

Course Title	Computer Networking Practice	Course No	To be filled by the office		
Specialization	Computer Engineering	Structure (IPC)	0	3	2
Offered for	UG	Status (Core / Elective)	Core		
Prerequisite	----	To take effect from			
Course Objectives	To understand basic networking commands, MAC/IP addressing, file transfer between two systems, etc. Simulation of error control techniques and flow control techniques using well-known protocols would be addressed as part of this course.				
Course Outcomes	Learner would be comfortable in design, testing, and trouble shooting aspects associated with local area networking. Learner would also appreciate the importance of error detecting codes and flow control techniques.				
Contents of the course	Connecting two nodes using Ethernet cable and study the performance evaluation parameters such as delay, effective bandwidth - Basic Networking commands – Ping, IPConfig, Traceroute, NSlookup - Introduction to Socket Programming. File transfer using TCP. Echo, Chat between two or more clients using socket programming - Simulation of Stop and Wait Protocol - Simulation of Stop and Wait protocol with NACK, Modelling of ACK, NACK drops, etc., - Modelling and simulation of Sliding window protocol - Sliding window protocol with ACK/NACK drops, frame drops etc., - Performance evaluation through simulation of IEEE 802.3/802.5 networks - Implementation of OSPF. Introduction to NS2/OPNET simulator, Case studies.				
Textbook	1. Larry L.Peterson and Bruce S Davie, Computer Networks: A systems Approach, 3rd Edn, Morgan, 2003. 2. William Stallings, Data and Computer Communications, 6th Edn, Pearson, 2000.				
References	1. Andrew S. Tanenbaum, Computer Networks, 4th Edn, 2003				

Course Title	Operating Systems Practice	Course No	To be filled by the office		
Specialization	Computer Engineering	Structure (IPC)	0	3	2
Offered for	UG	Status (Core / Elective)	Core		
Prerequisite	----	To take effect from			
Course Objectives	The course aims to equip the student with implementation level constructs / support in Linux for various concepts such as process management, concurrency, scheduling, deadlock avoidance, etc.				
Course Outcomes	The student shall be able to relate the operating system concepts listed above to the Linux operating system and support for the same available through various system calls.				
Contents of the course	Linux System Calls for process creation, management – Applications such as command prompt simulator using fork – Interprocess Communication using Shared Memory and Pipes – Producer Consumer – Applications using pipes / shm – Concurrency – Multithreading –Pthread support – Applications such as merge sort, min-max-average, etc. in a multi threaded fashion – Scheduling –pthread interfaces setschedpolicy – getschedpolicy based applications – Synchronization – threaded solution for classical problems like dining philosophers, readers writers, etc. using mutex locks and semaphores - Deadlock detection / avoidance algorithms.				
Textbook	1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, Operating System Concepts, John Wiley, 9th Edn, 2015.				
References	1. Robert Love, Linux Systems Programming, O Reilly Media, 2nd Edition 2. D Butlar, J Farrell, B Nichols, Pthreads Programming, O Reilly Media, 1996				

Course Title	Design for Quality and Reliability	Course No	To be filled by the office		
Specialization	Design	Structure (IPC)	3	0	3
Offered for	UG	Status (Core / Elective)	Core		
Prerequisite	Measurements and Data Analysis Lab (Probability and Statistics)	To take effect from			
Course Objectives	The objectives of the course are to help engineering students understand: (1) To understand concepts of quality & reliability (2) To evaluate the overall reliability of a system from component reliability.				
Course Outcomes	Attending the course would enable the student to: 1. Model repairable and non-repairable systems and calculate failure rate, repair rate, reliability and availability 2. Use various probability density distributions significant to reliability calculations 3. Fit a given failure data set of a product into a Weibull distribution and estimate the reliability parameters.				
Contents of the course	Module 1: Concepts of Product Quality • Quality Function Deployment / House of Quality • Six Sigma (6) Module 2: Concepts of Reliability · Basic concepts of repairable and non-repairable systems · Reliability, Availability and Maintainability (6) Module 3: Failure data analysis · Fitting discrete and continuous distributions to failure data sets, Weibull analysis, estimation of important reliability parameters (8) Module 4: Calculation of System Reliability from Component reliabilities · Markov modeling of repairable and non-repairable systems · Reliability Logic Diagrams · Fault-tree analysis (8) Module 5: Preventive and Predictive maintenance Failure Modes and Effects Analysis. (4)				
Textbook	1. Louis Cohen, Joseph P. Ficalora, *Quality Function Deployment and Six Sigma: A QFD Handbook*, Prentice Hall, Second Edition, 2009, ISBN: 9780137035441 2. VNA Naikan, *Reliability Engineering and Life Testing*, PHI Learning, 2010, ISBN: 978-8120335936 3. Singiresu S Rao, *Reliability Engineering*, Pearson Education, 2014, ISBN: 978-0136015727				
References	1. Patrick O Connor, *Practical Reliability Engineering*, John Wiley, Student ed., 2009, ISBN:9780470979815 2. 2. B.L. Hansen & P.M. Ghare, *Quality Control and Applications*, Prentice-Hall, 1997, ISBN: 9780137452255				

Course Title	Computer Architecture	Course No	To be filled by the office		
Specialization	Computer Engineering	Structure (IPC)	3	0	3
Offered for	UG	Status (Core / Elective)	Core		
Prerequisite	Computer Organization and Design	To take effect from			
Course Objectives	The course aims to expose students to the concepts involved in the design of computer systems covering aspects such as instruction sets, pipelining, caches, physical memory, virtual memory, superscalar and out-of-order instruction execution, vector processor and multi-threading				
Course Outcomes	Students will have the ability to design a computer system addressing issues related to Instruction level, data level and thread level parallelisms.				
Contents of the course	Fundamentals of Quantitative, Design and Analysis Computers. (3) Memory Hierarchy Design: Optimizations of Cache Performance, Memory Technology and Optimizations, Virtual Memory and Virtual Machines. (7) Instruction-Level Parallelism and Its Exploitation: ILP Concepts and Challenges, Overcoming Data Hazards with Static and Dynamic Scheduling, Reducing Branch Costs with Advanced Branch Prediction, Static and Dynamic Scheduling, Hardware-Based Speculation, Studies of the Limitations of ILP. (12) Multi-Threading: Exploiting Thread-Level Parallelism to Improve Uniprocessor Throughput (5) Data-Level Parallelism in Vector, SIMD, and GPU Architectures: Vector Architecture, Detecting and Enhancing Loop-Level Parallelism. (5) Thread-Level Parallelism: Centralized Shared-Memory Architectures, Performance of Symmetric Shared-Memory Multiprocessors, Distributed Shared-Memory and Directory-Based Coherence, Synchronization, Models of Memory Consistency, Multicore Processors and Their Performance. (5) Warehouse-Scale Computers to Exploit Request-Level and Data-Level Parallelism: Programming Models and Workloads for Warehouse-Scale Computers, Computer Architecture of Warehouse-Scale Computers, Physical Infrastructure and Costs of Warehouse-Scale Computers, Cloud Computing: The Return of Utility Computing. (5)				
Textbook	1. John L. Hennessy and David A. Patterson, Computer Architecture, Fifth Edition: A Quantitative Approach, The Morgan Kaufmann, 5th Edn, 2012.				
References	1. John P. Shen and Mikko H. Lipasti, Modern Processor Design: Fundamentals of Superscalar Processors, Waveland Press, 1st Edn, 2005, 2. D.M. Harris and S.L. Harris. Digital Design and Computer Architecture, 2nd Edn. Morgan Kaufmann, 2012. 3. M. Johnson. Superscalar Microprocessor Design, Prentice Hall, 1991.				

Course Title	Computer Architecture Practice	Course No	To be filled by the office		
Specialization	Computer Engineering	Structure (IPC)	0	3	2
Offered for	UG	Status (Core / Elective)	Core		
Prerequisite	----	To take effect from			
Course Objectives	The course aims to be a hands on to the supplementing theory course with exposure to issues related to computer systems design on instruction level ad thread level parallelism.				
Course Outcomes	Students will have the ability to design multi core systems for a given specification using electronic design automation tools.				
Contents of the course	Incrementally design, implement, test, and evaluate a complete multi-core system with an integrated collection of processors, memories. A processor includes – pipeline arithmetic operation, register file, branch predictors, hardware based instruction scheduling and commit, cache design, MESI.				
Textbook	1. John L. Hennessy and David A. Patterson, Computer Architecture, Fifth Edition: A Quantitative Approach, The Morgan Kaufmann, 5th Edn, 2012. 2. Samir Palnitkar, Verilog HDL: A Guide to Digital De sign and Synthesis, Second Edition, Prentice Hall, 2003.				
References	1. John P. Shen and Mikko H. Lipasti, Modern Processor Design: Fundamentals of Superscalar Processors, Waveland Press, 1st Edn, 2005, 2. D.M. Harris and S.L. Harris. Digital Design and Computer Architecture, 2nd Edn Morgan Kaufmann, 2012. 3. M. Johnson. Superscalar Microprocessor Design, Prentice Hall, 1991.				

Course Title	Artificial Intelligence	Course No *(will be assigned)*				
Specialization	Computer Science & Engineering	Structure (LTPC)	3	0	0	3
Offered for	UG	Status	Core ■		Elective ☐	
Faculty		Type	New ☐		Modification ☐	
Pre-requisite		To take effect from				
Submission date		Date of approval by Senate				
Objectives	The objective of this course is to train the students to understand different types of AI agents, various AI search algorithms, fundamentals of knowledge representation, building of simple knowledge-based systems and to apply knowledge representation, reasoning, and machine learning techniques to solve real-world problems.					
Contents of the course	Introduction: AI problems,Agents and Environments, Structure of Agents, Problem Solving Agents (4 hrs) Basic Search Strategies: Problem Spaces, Uninformed Search (Breadth-First, Depth-First Search, Depth-first with Iterative Deepening), Heuristic Search (Hill Climbing, Generic Best-First, A*), Constraint Satisfaction (Backtracking, Local Search) (7 hrs) Advanced Search: Constructing Search Trees, Stochastic Search, A* Search Implementation, Minimax Search, Alpha-Beta Pruning (7 hrs) Basic Knowledge Representation and Reasoning: Propositional Logic, First-Order Logic, Forward Chaining and Backward Chaining, Introduction to Probabilistic Reasoning, Bayes Theorem (7 hrs) Advanced Knowledge Representation and Reasoning: Knowledge Representation Issues, Non-monotonic Reasoning, Other Knowledge Representation Schemes (5 hrs). Reasoning Under Uncertainty: Basic probability, Acting Under Uncertainty, Bayes' Rule, Representing Knowledge in an Uncertain Domain, Bayesian Networks (6 hrs) Basic Machine Learning: Forms of Learning, Decision Trees, Nearest Neighbor Algorithm, Statistical-Based Learning such as Naïve Bayesian Classifier. (8 hrs)					
Textbook	Russell, S. and Norvig, P, *Artificial Intelligence: A Modern Approach*, Third Edition, Prentice-Hall, 2010					
References	1. Artificial Intelligence, Elaine Rich, Kevin Knight, Shivasankar B. Nair, The McGraw Hill publications, Third Edition, 2009 2. George F. Luger, Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Pearson Education, 6th ed., 2009. 3. Artificial Intelligence a new synthesis: Nils J. Nilson, Morgan Kaufmann Publishers, 1998					

Course Title	Embedded Systems Practice	Course No	To be filled by the office		
Specialization	Computer Engineering	Structure (IPC)	0	3	2
Offered for	UG	Status (Core / Elective)	Core		
Prerequisite	----	To take effect from			
Course Objectives	In this course fundamental practices in the context of embedded systems will be covered. Hands-on experiments will be performed involving TI ARM Cortex-M microcontroller LaunchPad IDE (and booster packs), rapid prototyping of embedded systems using open source microcontrollers (Arduino, Raspberry Pi, BeagleBone Black), wireless networked embedded systems using Arduino shields, and Internet of Things concepts such as smart automation.				
Course Outcomes	At the end of the course, a student will be able to, 1. Understand how embedded systems interfaces operate (GPIO, interrupts, ADC/DAC, etc.) using the ARM Cortex LaunchPad IDE and booster packs 2. Perform experiments in sound, video (gaming) and mobile robots, with LCD displays, stepper and DC motors and RC servos 3. Rapid prototype embedded systems using open source microcontrollers (such as Arduino, Raspberry Pi, BeagleBone Black, and Intel Edison/Galileo). 4. Build wireless networked embedded systems using Arduino shields and modules (e.g., GPS, GSM/GPRS, Bluetooth, RFID, and ZigBee). 5. Conduct experiments in Internet of Things (e.g., using Arduino Yun, Intel and Microsoft Developer Kits)				
Contents of the course	Experiments in GPIO, serial interfacing, interrupts, data acquisition with ADC, sound and video, DAC Experiments in control of RC servos, stepper motors, DC motors, and design of video games and mobile robots Data acquisition and real-time control with Arduino, Raspberry Pi, and BeagleBone Black microcontrollers, shields, and add-on boards Experiments in wireless networked systems, using shields and modules, for GPS, GSM/GPRS, ZibBee, Bluetooth, and RFID Experiments in IOT for smart automation, with Intel and Microsoft development kits				
Textbook	1. IIITDM Kancheepuram –Embedded Systems Practice Manual.				
References	1. Jonathan Valvano and Ramesh Yerraballi, 2014, “Embedded Systems – Shape the World” (ebook). 2. T. Igoe, 2007, “Making things talk”, O’Reilly Press.				

Course Title	Artificial Intelligence Practice	Course No *(will be assigned)*				
Specialization	Computer Science & Engineering	Structure (LTPC)	0	0	3	2
Offered for	UG & DD	Status	Core ■		Elective ☐	
Faculty		Type	New ☐		Modification ☐	
Pre-requisite		To take effect from				
Submission date		Date of approval by Senate				
Objectives	This course helps the students to get exposed to the use of AI-techniques such as searching, constraint satisfaction, knowledge representation and machine learning for solving classical AI-problems. With this practical exposure, the students will be able to build systems for solving real-world problems.					
Contents of the course	Solving travelling salesman problem using BFS, DFS and A* algorithms – Implementation of tic-tac-toe game problem using minimax algorithm with alpha beta pruning- Use Backtracking technique for solving the 8 queen problem- Develop the program for solving the 8 Puzzle problem using Hill Climbing Technique – Solve a simple time schedule problem using Constraint Satisfaction Procedure- Develop a simple question answering system by representing knowledge about a particular event- Develop disease identification system using Naïve Bayesian Classifier technique- Develop a decision tree classification system using C4.5 algorithm and extend your implementation to Random Forests and do the performance analysis - Implement K-nearest-neighbours (K-NN) algorithm for both classification and regression.					
Textbooks	1. Artificial Intelligence: Problems and their solutions, Danny Kopec, Shweta Shetty and Christopher Pileggi, Mercury Publishers, 2014. 2. Machine Learning for Big Data: Hands-On for Developers and Technical Professionals, Jason Bell, Wiley Publications, 2014.					
References	1. Russell, S. and Norvig, P, *Artificial Intelligence: A Modern Approach*, Third Edition, Prentice-Hall, 2010 2. Artificial Intelligence, Elaine Rich, Kevin Knight, Shivasankar B. Nair, The McGraw Hill publications, Third Edition, 2009					