# Current Courses

## Semester I (2017 Batch)

No course from CSE

## Semester III (2016 Batch)

### 1. Course code and title : CS2110 – Computer Programming lab

**2. Course category: PMT**

**3. Course credit : 0-0-2-2**

**4. Prerequisite course : No**

**5. Consent of teacher : No**

**6. Learning objectives :**

The aim of this course is to develop the basic programming abilities of students.

**7. Learning outcome:**

By the end of the course, students should be able to (1) **code proficiently in C** with a coherent coding style, (2)**debug code using a debugger** and make it a habit to use one, (3) **be comfortable with basic tools** viz. an editor, a debugger, makefiles, etc., (4)**be able to think in C** to the degree required to **implement pseudocode**.

**8. Syllabus**

* To build structures, and write libraries using the structures
* To write modular code and use Makefiles
* Recursion
* The basics of pointers, pointer arithmetic, arrays, and dynamic memory allocation
* Linked List , Stack Queue
* Sorting, Searching including tree search

Proposing Faculty :

Department / Centre : Computer Science and Engineering  
Programme : B.Tech

Proposal Type: Old

### 1. Course code and title :CS2100 - Discrete Mathematics for Computer Science

**2. Course category: PMT**

**3. Course credit : 3-0-0-3**

**4. Prerequisite course : No**

**5. Consent of teacher : No**

**6. Learning objectives :**

**7. Learning outcome:**

**8. Syllabus :**

*Introduction to Propositional Logic:* Logical Operators -- negation, conjunction, disjunction, XOR, conditional, biconditional. Precedence of logical operators. The conditional operator, examples of translating English sentences into compound propositions, logical equivalences, some examples of logic puzzles. Predicates, definitions, examples, Existential and Universal Quantifiers, Uniqueness quantification using existential and universal quantification, Restricting domain in case of existential and universal quantification. Logical Equivalences between predicates. Predicates with multiple variables. Examples. Nested Quantifiers. Order of Nesting of quantifiers. Examples where the order can be flipped and cases in which it cannot be. Thinking of quantifiers as loops.  
Nested Quantifiers, examples. Arguments, Argument Form. Valid arguments. Rules of Inference. Modus Ponens, Modus Tollens, Addition, Simplification. Why is a particular argument form valid or invalid? Rules of Inference. Arguments using these rules of inference. Examples.

*Proof Techniques*: Direct Proof, Proof by contradiction, proof by contraposition. Examples in each case. Pitfalls in proof by enumeration. Example. Some more examples of Proof by contrapositive, contradiction. Proof by cases, exhaustive proofs. Backward reasoning: examples arithmetic and geometric mean. Game of marbles. Existential proofs (non-constructive): examples. Chomp.   
Proof strategies: checkerboard and tilings. Examples.

*Introduction to Sets*: definitions, subsets, null-set, power-set, set operations, computer representation of sets. Russels paradox, barber's puzzle. Cardinality of sets, infinite sets, cardinality of infinite sets. Detour to Cartesian Products, relations, function, one-one, onto, one-one onto functions. Countably infinite sets, Set of integers is countable, set of positive rationals is countable, set of reals is uncountable (Cantor's diagonalization argument).

*Mathematical induction:* Why it works. Well ordering principle. Examples using induction and strong induction. Sequences.Well ordering principle, proofs using WOP, example on Tournaments and existence of GCD. Brief discussion on recursion and recursively defined objects. Trees. Recursion, recursive functions, height of a tree, number of nodes.

*Proving program correctness:* Pre-conditions, post-conditions, loop invariants. Program termination. Can termination be checked by a general algorithm? Discussion on the halting problem.

*Relations revisited:* Properties of relations. Reflexive, Symmetric, Antisymmetric, Transitive. Examples. Relations represented as matrices, diagraphs. closures of relations. Reflexive, symmetric, and transitive closures. Computing them. A naive method to compute transitive closure. Calculation of number of bit operations required for the algorithm. Warshall's algorithm for computing transitive closure. Calculation of number of bit operations for Warshall's algorithm. Equivalence relations, equivalence classes, partitions. Partial orders, examples, Hasse diagram, minimal, maximal elements, least and greatest elements, chains and antichains.

*Graphs*: Graphs as relations, directed graphs, undirected graphs, special types of graphs, paths, cycles, trees, cliques, bipartite graphs. Bipartite graphs continued, Matchings in bipartite graphs. Relation between Matching and Vertex cover. Halls theorem. Proof of Halls Theorem. Konig's theorem and proof of it using Hall's Theorem. Coloring of graphs. Bipartite graphs are 2-colorable. Existence of a K\_s implies that the chromatic number is greater than s. Greedy coloring. Introduction to Planar graphs. Planar graphs and properties -- Euler's formula, proof, bound on number of edges of a planar graph. Coloring of planar graphs using 6 colors. Statement of Kuratowaski's theorem (without proof). Definition of Homeomorphism.

*Basics of Counting:* Sum product rule. Pigeon Hole principle and applications of pigeon hole principle to various problems. Permutations and Combinations, Binomial theorem and identities. Proofs using combinatorial arguments. Permutations and combinations with repetition. Generating permutations and combinations. Algorithms for the same.

*Introduction to discrete probability:* Examples. Independence and conditional probability. Examples of independent events. Probabilistic reasoning. Monty Hall 3 Door puzzle. Random variables, expectation, linearity of expectation. Examples. Hatcheck Problem, Number of inversions in a permutation.

*Advanced Counting techniques*: Modelling problems with recurrences relations. Towers of Hanoi, Fibonacci, Catalan numbers. Catalan Numbers: Balanced Parenthesis, Number of rooted full binary trees, Diagonal Avoiding paths. Recursive formulation. Catalan Numbers -- proof of closed form formula. Equivalence to number of mountain ranges with up and down strokes and the number of ways to parentheize.  
  
8. **Textbook**

*Discrete Mathematics and Applications* by Kenneth Rosen (7th Edition, 2012), McGraw-Hill Education (ISBN-13: 978-0073383095)

*Discrete Mathematics* by Norman L. Biggs (2nd Edition, 2002), Oxford University Press (ISBN-13: 978-0198507178)

Proposing Faculty :

Department / Centre : Computer Science and Engineering  
Programme : B.Tech

Proposal Type: Old

## Semester V (2015 Batch)

### 1.Course code and title: CS3300 - Compiler Design

**2.Course category**: PMT

**3.Course credit** : 3-0-0-3

**4.Prerequisite course**:

CS2200 : Languages, Machines and Computation

CS2800 : Data Structures Algorithms

**5.Consent of teacher** : No

**6. Learning objectives :**

A compiler is program that convert high level (human understandable) language to low level (machine understandable) code. Compilers are what makes writing modern software possible and their study has been one of the classical topics in an Undergraduate Computer Science curriculum. The objective of the course is to introduce the basic theory

underlying the different components and phases of a compiler like parsing, code generation etc. Simultaneously, we familiarise the students to the various tools that are

used for building modern compilers.

**7. Learning outcome:**

At the end of the course we expect the student to know enough of the theory (Parsing, Code generation, optimisation) and tools (parser generators, code generators) so that they can build a compiler that converts from a non-trivial high level language to machine code.

**8. Syllabus :**

* Introduction to language translators and overview of the compilation process.
* Lexical analysis: specification of tokens, token recognition, conflict resolution.
* Parsing: Overview of CFG, Parse trees and derivations, left recursion, left factoring, top-down parsing, LALR parsing, conflict resolution, dangling-else.
* Syntax directed translation. Semantic analysis, Type checking, intermediate code generation.
* Runtime environments: activation records, heap management
* Code optimization: basic blocks, liveness, register allocation.
* Advanced topics: Overview of machine dependent and independent optimizations.

**Textbook(s)**

* Compilers: Principles, Techniques, and Tools, Alfred Aho, Monica Lam, Ravi Sethi, Jeffrey D. Ullman, Addison-Wesley, 2007
* Modern Compiler Implementation in ML, Andrew Appel, Cambridge University Press (8 July 2004), ISBN-13: 978-0521607643

**References**

Compiler Construction: Principles and Practice, 1st Edition, Kenneth C. Louden, Cengage Learning; 1 edition (January 24, 1997), ISBN-13: 978-0534939724

Proposing Faculty : Piyush Kurur

Department / Centre : Computer Science and Engineering

Programme : B.Tech

Proposal Type: Name change from old IITM syllabus to new.

### 1.Course code and title : CS3310 - Compiler Design Lab

**2. Course category: PML**

**3. Course credit : 0-0-2-2**

**4. Prerequisite course :**

* CS2810 : Data Structures Algorithms Lab.

**6. Syllabus**

This is a companion lab course forCS3300 Compiler design course. The Lab part shall include experiments that shall illustrate various tools and like ml-yacc, ml- lex that are used to build a modern compiler.

**5. Consent of teacher : No**

Proposing Faculty : Piyush Kurur

Department / Centre : Computer Science and Engineering

Programme : B.Tech

Proposal Type: Name change from old IITM syllabus to new.

### 1. Course code and title : CS3100, Paradigms of Programming

**2. Course category: PMT**

**3. Course credit : 3-0-0-3**

**4. Prerequisite course : No**

**5. Consent of teacher : Yes**

**6. Learning objectives :**

To learn different paradigms of programming, their similarities and differences, and to be able to choose a paradigm based on the objecting of programming.

**7. Learning outcome:**

To be able to choose a paradigm based on the objecting of programming

**8. Syllabus:**

* Introduction to different paradigms of programming: Imperative - Object Oriented - Functional - Logic Imperative and Object-oriented Programming - Role of Types - Static and Dynamic Type Checking - Scope rules ; Grouping Data and operations, Information Hiding and Abstract Data Types, Objects, Inheritance, Polymorphism,Templates.
* Functional Programming - Expressions and Lists, Evaluation, types, type systems, values and operations, function declarations, lexical scope, lists and programming with lists, polymorphic functions, higher order and Curried functions, abstract data types.
* Logic Programming - Review of predicate logic, clausal-form logic, logic as a programming language, Unification algorithm, Abstract interpreter for logic programs, Semantics of logic programs, Programming in Prolog.

**9.Textbook:**

* Programming Languages: Principles and Paradigms   
  Authors: Maurizio Gabbrielli   
  Publisher: Springer; 2010 edition (15 April 2010)  
  ISBN-13: 978-1848829138
* Programming Languages: Concepts & Constructs  
  Authors: Ravi Sethi  
  Publisher: Pearson Education; 2 edition (2006)  
  ISBN-13: 978-8177584226
* Programming Languages: Design and Implementation  
  Authors: Terrence W. Pratt (Author), Marvin V. Zelkowitz (Author)  
  Publisher: Pearson; 4 edition (28 August 2000)  
  ISBN-13: 978-0130276780
* Programming Language Pragmatics  
  Authors: Scott Michael L.   
  Publisher: Elsevier India (2014)  
  ISBN-13: 978-8131222560

Proposing Faculty : MKD

Department / Centre : Computer Science and Engineering

Programme : B.Tech

Proposal Type: Old

### 1. Course code and title : CS3500 - Operating Systems

**2. Course category: PMT**

**3. Course credit : 3-0-0-3**

**4. Prerequisite course :** CS2800 : Data Structures and Algorithms

CS2600 : Computer Organization and Architecture,

CS2610 : Computer Organization and Architecture Lab

**5. Consent of teacher : No**

**6. Learning objectives :** This course will introduce the student to the basic concepts involved in the design and implementation of an operating system. Students will be made familiar to the important modules of operating systems like process management, memory management, file systems, synchronization primitives and exception handling. Important data structures used in the design of these modules will be introduced. The accompanying lab course CS3510 is intended to give students an illustration of the concepts introduced in the theory course.

**7. Learning outcomes :** Students will be able to develop an understanding of how an operating functions as a middle layer between the hardware of a computer and the user programs and the various tasks involved in this. By the end of this course, students will be able to appreciate the design issues and concepts underlying some of the well known operating systems and compare the pros and cons of various design options and various issues involved in the design of a large software.

**8. Syllabus**

* Process Management : Scheduling (essential topics: context-switch, unix fork, scheduling algorithms representing fairness, infinite wait, optimal scheduling, priority inversion)
* Synchronization Primitives and Problems, Deadlocks (essential topics: Peterson's algorithm, monitors)
* Memory Management : Virtual Memory, Demand Paging (essential topics: fragmentation, pinning, Belady's anomaly, thrashing)
* File systems; I/O Management (essential topics: DMA, delayed writes, elevator algorithm). Security.

**9. Text Book:**

1. Charles Crowley, Operating Systems: A Design-Oriented Approach, International edition, McGraw-Hill Education (ISE Editions), ISBN-13: 978-0071144629

2. Abraham Silberschatz, Peter B. Galvin and Greg Gagne, Operating Systems Concepts, Wiley, 2015, ISBN-13: 978-8126554270

**10. Reference Book:**

1. Andrew S. Tanenbaum, Herbert Bos, Modern Operating Systems, Pearson Education India; Fourth edition 2016, ISBN-13:978-9332575776
2. William Stallings, Operating Systems: Internals and Design Principles, Pearson Education India; 7 edition (2013), ISBN-13: 978-9332518803
3. Robert Love, Linux Kernel Development (Developer's Library), Pearson Education India, 3 edition (2010), ISBN-13: 978-8131758182

Proposing Faculty :

Department / Centre : Computer Science and Engineering

Programme : B.Tech

Proposal Type: Old

### 1. Course code and title : CS3510 - Operating Systems Lab

**2. Course category: PML**

**3. Course credit : 0-0-2-2**

**4. Prerequisite course :** CS2810 : Data Structures and Algorithms Lab

CS2600 - Computer Organization and Architecture,

CS2610 - Computer Organization and Architecture Lab

**5. Consent of teacher : Yes**

**6. Syllabus**

This is a companion lab course forCS3500 Operating Systems course. The Lab part shall include experiments that shall illustrate booting of a system, implementation of process, memory, file and I/O management concepts, preferably using a popular operating system as a case study.

Proposing Faculty :

Department / Centre : Computer Science and Engineering

Programme : B.Tech

Proposal Type: Old

### 1. Course code and title : CS4801 - Principles of Machine Learning

**2. Course category: PME/PME\*/GCE**

**3. Course credit : 3-1-0-4**

**4. Prerequisite course : No**

**5. Consent of teacher : Yes**

**6. Learning objectives :**

This is an intermediate level course in computer science field and assumes background in algorithm, programming and introductory knowledge of probability and linear algebra. The main objective of the course is to introduce student with methodologies and applications of various topics of machine learning. The course will also provide sufficient background to motivate students to take up advanced levels courses related to machine learning, deep learning, bioinformatics, robotics, Artificial intelligence etc.

**7. Learning outcome:**

Upon successful completion, the students will be able to implement few machine learning models like regression classification and clustering and will be able to learn models from data and evaluate models on test data.

**8. Syllabus**

* Introduction to the course, recap of linear algebra and probability theory basics.
* Linear Regression, Ridge Regression, Sensitivity Analysis, Multivariate Regression.
* Bayesian Classification: Naive Bayes, Parameter Estimation (ML, MAP), Sequential Pattern Classification.
* Evaluation and Model Selection: ROC Curves, Evaluation Measures, Significance tests.
* Non-parametric Methods: k-Nearest Neighbours, Parzen Window.
* Discriminative Learning models: Logistic Regression, Perceptrons, Artificial Neural Networks, Support Vector Machines.
* Dimensionality Reduction: Principal Component Analysis, Fischer's Discriminant Analysis.
* Decision Trees: Splitting Criteria, CART.
* Ensemble Methods: Boosting, Bagging, Random Forests.
* Clustering: Partitional, Hierarchical, density based clustering.

**9. Text Book:**

* Christopher Bishop. [Pattern Recognition and Machine Learning](http://research.microsoft.com/en-us/um/people/cmbishop/prml/). ISBN 0387310738. (free online)
* Trevor Hastie, Robert Tibshirani, Jerome Friedman.[Elements of Statistical Learning](http://www-stat.stanford.edu/~tibs/ElemStatLearn/). ISBN 0387952845. (free online)
* Richard Duda, Peter Hart, David Stork, [Pattern Classification](http://cns-classes.bu.edu/cn550/Readings/duda-etal-00.pdf), 2nd Ed.,, John Wiley & Sons, 2001. ISBN 9788126511167

**10. Reference Book:**

Tom Mitchell. [Machine Learning](http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/mlbook.html). ISBN 0070428077.

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. Introduction to Statistical Learning, Springer, 2013. ISBN 9781461471387

Anand Rajaraman, Jurij Leskovec, and Jeffrey Ullman. [Mining of Massive Datasets.](http://infolab.stanford.edu/~ullman/mmds.html) Cambridge University Press. 2012. (free online)

Shai Shalev-Shwartz, and Shai Ben-David, [Understanding Machine Learning: From Theory to Algorithms](http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/index.html), Cambridge University Press, 2014. (Can be downloaded as PDF file.)

* Sam Roweis's [probability review](http://cs.nyu.edu/~dsontag/courses/ml12/notes/probx.pdf)
* Sam Roweis's [linear algebra review](http://cs.nyu.edu/~dsontag/courses/ml12/notes/linear_algebra.pdf)

[Convex Optimization](http://www.stanford.edu/~boyd/cvxbook/) by Stephen Boyd and Lieven Vandenberghe. (Can be downloaded as PDF file.)

Proposing Faculty : Dr Sahely Bhadra  
Department / Centre : Computer Science and Engineering  
Programme : B.Tech 5th semester PME for CS and CGE for the rest

Proposal Type: Exactly same as IITM CS4011 course which in under approval in IITM.

### 1. Course code and title : CS4803 Model Checking

**2. Course category: PME/PME\* for Computer Science**

**3. Course credit: 3-1-0-4 (L-T-P-C)**

**4. Prerequisites:** Familiarity with basic algorithms and finite-state machines

**5. Consent of teacher : Yes**

**6. Learning Objectives:**

In today’s world, many safety-critical systems are automatically controlled by an embedded code (automatic cars, aircrafts, pacemakers, etc). As the size and the number of interacting components increase, the design and verification of these control software becomes increasingly complex and highly error-prone. Model checking is a field of research that addresses this challenge by making use of rigorous mathematical techniques. The key idea is to view control software as extensions of finite-state machines - this is the “model” of the software. Verification of the software translates to “checking” if the model satisfies certain properties. This is done by an algorithmic analysis of the model.

**7. Learning outcome:**

The goal of this course is to understand the theoretical foundations of model Checking technology and get a hands-on experience with a tool that performs model checking.

**8. Syllabus:**

*Module 1: Modeling software and hardware as finite state machines (automata):*

Models for simple control code and hardware circuits as transition systems; properties as formal languages; introduction to NuSMV model-checker

*Module 2: Automata theory over finite and infinite words:*

Review of DFAs, NFAs, closure properties; introduction to Büchi automata, non-deterministic and deterministic Büchi automata, closure properties.

*Module 3: Temporal logics for specifying properties:*

Linear Temporal Logic (LTL), converting LTL to Büchi automata, model-checking LTL on transition systems, Computation Tree Logics (CTL and CTL\*), expressiveness of LTL, CTL and CTL\*, model-checking CTL on transition systems

*Module 4: Approaches for scaling model-checking to real systems:*

Binary decision diagrams (BDDs), Satisfiability solving based model-checking algorithms

*Module 5: Adding time to automata:*

Modeling systems with timing constraints; introduction to timed automata

**9. References:**

*Principles of Model-checking*, Christel Baier and Joost-Pieter Katoen, MIT Press (2008).

ISBN-13: 978-0262026499

*Decision Procedures*, Daniel Kroening and Ofer Strichman, Springer (2ed, 2016) ISBN-13: 978-3662504963

Proposing Faculty : Deepak Rajendraprasad for Dr. B. Srivathsan (CMI)

Department / Centre : Computer Science and Engineering  
Programme : B.Tech

Proposal Type: New course

### 1. Course code and title : CS5820 - Probability and Computing

**2. Course category: PME/PME\* for Computer Science**

**3. Course credit: 3-1-0-4 (L-T-P-C)**

**4. Prerequisites:**

**5. Consent of teacher : Yes**

**6. Learning Objectives:**

To introduce the power of probability theory and randomization techniques in computer science, with particular emphasis on analyzing algorithms that employ randomization.

**7. Learning outcome:** Students will be able to model probabilistic events using random variables and analyze simple probabilistic algorithms using probabilistic tools acquired in this course.

**8. Syllabus:**

*Introduction*: Events, probability spaces, random variables, expectation, conditional expectation, tail bounds including Markov's inequality, Chebyshev's inequality, Chernoff bounds. Sample applications include Karger's min-cut algorithm, randomized quicksort, and permutation routing on the hypercube.

*Common Probability Distributions*: Bernoulli and binomial random variables, geometric distribution, coupon collector's problem, Poisson distribution, normal distribution, power law distributions.

*Hashing with Applications*: Balls into bins, chain hashing, Bloom filters, pairwise independence, Chebyshev's inequality for pairwise independent variables, universal hash functions, perfect hashing, the count-min sketch, the power of two choices, cuckoo hashing.

*Probabilistic Method*: The counting argument, the expectation argument, sample and modify, the second moment method, the conditional expectation inequality, the Lovasz local lemma.

*Markov Chains and Random Walks*: Basic definitions, stationary distribution, variation distance and mixing time and their relation to graph spectrum, random walks on undirected graphs, the Monte Carlo method, the Metropolis algorithm, coupling.

*Martingales*: Basic definitions, stopping time, Wald's equation, Azuma-Hoeffding inequality.

**9. References:**

1. Randomized Algorithms, by Motwani and Raghavan, Cambridge University Press, 1995, **ISBN-13:** 978-0521474658.
2. Concentration of Measure for the Analysis of Randomized Algorithms, by Dubhashi and Panconesi, Cambridge University Press, 2009, ISBN: 978-1107606609
3. The Probabilistic method, by Alon and Spencer, 3rd edition, Wiley, 2008, ISBN 978-0-470-17020-5.

Proposing Faculty :

Department / Centre : Computer Science and Engineering

Programme : B.Tech

Proposal Type: Old

# Other Courses

## Semester II(2017 Batch)

### 1.Course Code and Title :CS1100 Introduction to Programming

**2.Course Category :** BET  
**3.Course Credit:** 2-0-2-4

**4.Prerequisite Course:** Nil

**5.Consent of Teacher :** No

**Learning Objectives:**

No prior programming experience is assumed. The main objective is to develop the skill to solve simple computational problems by designing step by step logical solutions and converting them to computer programs. Standard programming constructs like conditional execution, iteration, arrays and functions will be introduced. Basics of organization of a computing unit and computer representation of numbers will also be covered. Towards the last part of the course students will be introduced to data abstraction. This course will provide students with sufficient coding skills for any course with a programming component like Programming and Data Structures, Computational methods and CAD Laboratory.

**Learning Outcomes:**

By the end of the course, students will be able to design and code simple moderate sized (100 to 200 lines) programs for solving simple computational tasks. Students will also be able to understand and debug moderate sized programs written by others. Through the lab exercises, students will be able to write modular and maintainable programs, following one of the standard coding conventions.

**Brief Syllabus:**

Basic organization of a computer: ALU, input-output units, memory, program counter - variables and addresses - instructions: store, arithmetic, input and output - simple sequential programs - memory state transitions - conditional instruction: if then else, control flow diagrams, nested conditions, iterations: while loop and its control flow, arrays: indexing, memory model, programs with array of integers, two dimensional arrays - Variants of conditional and iterative instructions – pointers: dereferencing and address operators, pointer and address arithmetic, array manipulation using pointers - functions: modularity, declaration and definition, function call and return and associated control flow, functions with parameters, returning a value, multiple parameters, data flow during function call, modifying parameters inside functions using pointers, arrays as parameters. Computer representation of integers, signed and unsigned integers, ranges and overflow, Different data types - floating point representations, precision and range, error in representation, functions with different parameter types and return types, user defined data types for data abstraction.

Recommended programming language to be used for instruction is C.

**Text books** :

1. Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language*, Prentice Hall, Second Edition, 1988. ISBN:9780131103627.
2. Byron S Gottfried, *Schaum's Outline of Programming with C* (Schaum's Outline Series), McGraw-Hill Education, 1996, ISBN:9780070240353

Proposing Faculty : Dr. Jasine Babu  
Department / Centre : Computer Science and Engineering  
Proposal Type : Replacement course for CS1100 Computational Engineering

Programme : B.Tech (all branches). This course is already approved by BAC and Senate of IITPKD

## Semester IV

## 

## Semester VI

### 1.Course code and title : CS4802 Design and Analysis of Algorithms

**2.Course category:** PME/PME\*

**3.Course credit:** 3-1-0-4 (L-T-P-C)

**4.Prerequisite course:** Discrete Mathematics for Computer Science, Data Structures and Algorithms.

**5.Consent of Teacher :** yes

**6. Learning Objectives:** This is an intermediate level course on algorithm design techniques. The first part of this course course will provide a detailed introduction to different algorithm design paradigms .

The middle part of this course is intended to make students familiar with some well known classical algorithms and their design strategies and efficiency analysis. The third part of this course will deal with computationally hard problems and tacking them using approximation algorithms.

7. Learning Outcomes: By the end of this course, students will be able to design efficient algorithms for moderately difficult computational problems, using various algorithm design techniques taught in the course. Students will be able to apply commonly used approximation techniques like randomization and LP rounding in computationally hard problems, yet simple to describe optimization problems.

8. Syllabus:

Greedy Algorithms: Greedy choice, Example like Interval Scheduling, Fractional knapsack, and Huffman coding.

Dynamic Programming: Optimal Substructure and Overlapping sub problems, Examples like Weighted Interval Scheduling, Integral knapsack, Coin changing and Longest increasing subsequence.

Network Flow algorithms: Max-flow mincut theorem, Ford Fulkerson algorithms, Edmond's shortest path heuristic, Dinitz algorithm, Bipartite Matching.

Matching: Weighted Bipartite Matching, Hungarian algorithm, Edmond's algorithm for matching in general graphs.

String Matching: KMP algorithm, Boyer Moore algorithm

NP-completeness: Reduction amongst problems, classes NP, P, NP-complete, and polynomial time reductions

Approximation algorithms: Approximation factor, Vertex cover via maximum matching, greedy algorithm for set cover, Metric TSP, Hardness of approximation, LP rounding – set cover, vertex cover. Randomized approximation - Max-Cut, De-randomization using conditional expectation.

9. Textbooks :

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L Rivest, Clifford Stein, Introduction to Algorithms, Third Edition, PHI Learning, 2009, ISBN:978-81-203-4007-7.
2. Jon Kleinberg and Eva Tardos, Algorithm Design, Pearson, 2015, ISBN:978-93-325-1864.
3. Sanjoy DasGupta, C. H. Papadimitriou, Umesh Vazirani, Algorithms, First Edition, Tata McGraw Hill, 2006, ISBN: 978-0073523408.

10. References :

1. Dexter C Kozen, The Design and Analysis of Algorithms, Texts and Monographs in Computer Science, Springer, 1992, ISBN:0-387-97687-6.

Proposing Faculty : Dr. Jasine Babu

Department / Centre : Computer Science and Engineering

Proposal Type : Change in code (already approved in BAC and Senate with a different course code)

Programme : B.Tech (Computer Science and Engineering)

## 

# 2015 and 2016 Batches

## Semester I

### 1. Course code and title : CS1010 Computational Engineering

**2. Course category: BET**

**3. Course credit : 3-0-1-4**

**4. Prerequisite course : No**

**5. Consent of teacher : Yes**

**6. Learning objectives :**

The course aims to provide exposure to problem-solving through programming. It aims to train the student to the basic concepts of the C-programming language. This course involves a lab component which is designed to give the student hands-on experience with the concepts.

**7. Syllabus**

* Module 1 : (Introduction to Computing) - 6 lectures
  + Fundamentals of Computing, Historical perspective, Early computers. Computing machine. Problems, Pseudo-code and flowcharts. Memory, Variables, Values, Instructions, Programs.
* Module 2 : (Introduction to C) 10 lectures
  + The language of C : Phases of developing a running computer program in C.
  + Data concepts in C :Constants, Variables, Expressions, Operators, and operator precedence in C.
  + Statements : Declarations, Input-Output Statements, Compound statements, Selection Statements. Conditions, Logical operators, Precedences. Repetitive statements, While construct, Do-while Construct, For construct.
  + Data types, size and values. Char, Unsigned and Signed data types. Number systems and representations. Constants, Overflow.
  + Arrays. Strings. Multidimensional arrays and matrices.
* Module 3 : (Modular Programming and Example Problems) ​: 10 lecture
  + Functions :The prototype declaration, Function definition.
  + Function call : Passing arguments to a function, by value, by reference. Scope of variable names. Recursive function calls, Tail recursion. Analysing recursion, Tree of recursion, linear recursion.
  + Sorting problem : Selection Sort, Insertion Sort, Comparison between sorting algorithms. Sorting in multidimensional arrays. Sorting in strings.
  + Search problem : Linear search and binary search. Comparison between search procedures. Recursive and Iterative formulations.
* Module 4 : (More Data Types in C) 14 lectures
  + Pointers : Pointer variables. Declaring and dereferencing pointer variables. Pointer Arithmetic. Examples. Accessing arrays through pointers. Pointer types, Pointers and strings. String operations in C.
  + Structures in C : Motivation, examples, declaration, and use. Operations on structures. Passing structures as function arguments. type defining structures.
  + Self-referential structures. Dynamic Data Structures. Linked Lists. Examples.
  + File input-output in C. Streams. Input, output and error streams. Opening, closing and reading from files. Programming for command line arguments.
  + Numerical errors due to data representations and machine precision. Approximation and error analysis. Illustration through examples.

**8. Text Book:**

* C: How to program, H. M. Deitel, P. J. Deitel, 7th edition, Pearson Education, 2010.
* R. G. Dromey, "How to Solve It By Computer", Pearson, 1982
* A.R. Bradley, "Programming for Engineers", Springer, 2011
* Kernighan and Ritchie, "The C Programming Language", (2nd ed.) Prentice Hall, 1988

Proposing Faculty :   
Department / Centre : Computer Science and Engineering  
  
Programme : B.Tech

Proposal Type:

## Semester II

no courses

## Semester IV

### 1. Course code and title : CS2200 - Languages, Machines and Computation

**2. Course category: PMT**

**3. Course credit : 3-1-0-4**

**4. Prerequisite course : No**

**5. Consent of teacher : Yes**

**6. Learning 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.

**7. Syllabus**

Four basic themes (but related in a flow) :

* Finite Automata & Regular Languages  
  Languages vs Problems. Finite State Automata, Regular Languages. Closure properties, Limitations, Pumping Lemma, Myhill-Nerode relations, Quotient Construction. Minimization Algorithm.
* 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.
* 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.
* 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.

**8. TextBook**

* Automata and Computability, Dexter C. Kozen, Springer Publishers, 2007.
* Introduction to Automata Theory, Languages and Computation, Hopcroft, Motwani, and Ullman, Pearson Publishers, Third Edition, 2006.

**9. References:**

* Elements of the Theory of Computation, H. R. Lewis and C.H. Papadimitriou, Prentice Hall Publishers, 1981
* Introduction to Languages and the Theory of Computation, John. C. Martin, Tata McGraw-Hill, 2003.

Proposing Faculty :   
Department / Centre : Computer Science and Engineering  
Programme : B.Tech

Proposal Type: Old

### 1.Course code and title : CS2800 Data Structures and Algorithms

**2.Course category: PME/PME\***

**3.Course credit: 3-1-0-4 (L-T-P-C)**

**4.Prerequisite course: CS1100 Computational Engineering**

**5.Consent of Teacher : no**

**6. Learning Objectives:**

1. Acquire some basic tools and techniques of algorithm analysis
2. Learn about some important data structures that enables one to choose the appropriate data structure for designing efficient algorithms.
3. Learn some basic algorithms and their analysis.

**7. Learning Outcomes:** By the end of this course, students will be able to choose an appropriate data structure to model simple computational problems in hand so as to make the solution work efficiently from a computational point of view. They will be able to compute the asymptotic complexity of their algorithm.

**8. Syllabus:** Abstraction - Abstract data types; Data Representation; Elementary data types; Basic concepts of data Structures; Mathematical preliminaries - big-Oh notation; efficiency of algorithms; notion of time and space complexity; performance measures for data structures.

ADT array - Computations on arrays - sorting and searching algorithms.

ADT Stack, Queue, list - array, linked list, cursor based implementations of linear structures. ADT Tree - tree representation, traversal of trees; ADT Binary tree - binary trees, threaded binary trees, application of binary trees - Huffmann coding; application of threaded binary trees - differentiation;

Search Tree - Binary search tree; balanced binary search trees - AVL tree; Applications of Search Trees - TRIE; 2-3 tree, 2-3-4 tree; concept of B-Tree. ADT Dictionary - array based and tree based implementations; hashing - definition and application - LZW encoding. ADT Priority Queue - Heaps; heap-based implementations; applications of heaps - sorting; Graphs - shortest path, minimum spanning tree, DFS, BFS - an application of DFS and BFS. Algorithm Design Paradigms - greedy, divide and conquer, dynamic programming, backtracking.

9. Textbooks :

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L Rivest, Clifford Stein, *Introduction to Algorithms*, Third Edition, PHI Learning, 2009, ISBN:978-81-203-4007-7.
2. Sanjoy DasGupta, C. H. Papadimitriou, Umesh Vazirani, *Algorithms,* First Edition, Tata McGraw Hill, 2006, ISBN: 978-0073523408.
3. Jon Kleinberg and Eva Tardos, *Algorithm Design*, Pearson, 2015, ISBN:978-93-325-1864.

10. References :

1. Dexter C Kozen, *The Design and Analysis of Algorithms*, Texts and Monographs in Computer Science, Springer, 1992, ISBN:0-387-97687-6.

### 1.Course code and title : CS2810 Data Structures and Algorithms Laboratory

**2.Course category: PML**

**3.Course credit: 0-0-2-2 (L-T-P-C)**

**4.Prerequisite course: CS1100 Computational Engineering**

**5.Consent of Teacher : no**

**6.Syllabus:** This is a companion lab course forCS2800 : Data Structures and Algorithms course. The laboratory component will require the student to write computer programs using a careful choice of data structures (in C language) from scratch, based on the concepts learnt in the theory course.