Birla Institute of Technology & Science, Pilani
Pilani Campus

INSTRUCTION DIVISION
SECOND SEMESTER 2015-2016
Course Handout - Part II

In addition to part-I (General Handout for all courses appended to the time table) this portion gives further specific details regarding the course

Course No. : **CS F364**
Course Title **:** **Design & Analysis of Algorithms**

Instructor-in-Charge: **Shan SundarBalasubramaniam (email: sundarb)**
Instructor(s): **JagatSeshChella (email: jagatsesh)**

***Course Website: access via*http://nalanda.bits-pilani.ac.in**

1. a. Course Objective:

The objective of the course is to impart students

- a working knowledge of how to choose and apply one of several Algorithm Design techniques to solve a given (computational)problem and
- the ability to study, analyze, and understand a given problemso as to be able to formulate it computationally, relate it to other known problems, and characterize the difficulty in solving it.

1.b. Course Scope

This course will cover the broad subject of Algorithms with emphasis on Design Techniques under the assumption that the student has already learnt the basics of data structures and algorithms. The scope of the course would broadly include the following:

- **Problems** (Types of problems, Formulation of problems, and Reductions between problems; Specific Problem Domains such as Graphs, Text Processing, and Number Theory)
- **Algorithm Design** (Basic techniques – Divide & Conquer, Dynamic Programming, Greedy, and Randomization; Techniques for handling hard problems – Backtracking, Branch & Bound, and Approximation Algorithms).
- **Complexity Theory** (Lower Bounding of complexity of problems, Complexity Classes including Approximation Classes, and NP-completeness)

2. Text and References:

a. Text Book:

T1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms.* MIT Press. 3rd Edition.

b. References:

Birla Institute of Technology & Science, Pilani
Pilani Campus, VidyaVihar
Pilani 333031, Rajasthan, India

Tel: +91 1596 245073
Fax: +91 1596 244183
Web: www.pilani.bits-pilani.ac.in

R1. Ellis Horowitz, SartajSahni and SanguthevarRajasekaran. *Computer Algorithms.* Universities Press. 2007.

AR. *Additional references to be posted on the course website where applicable*.

3. Course Plan:

3. a. Pre-requisite Topics:
During the course it will be assumed that the student has systematically studied and understood the following topics:

S.No.	Topics	Nature and level of understanding required and assumed
1	Discrete Structures	[***Basic***]Properties and theoretical results on **Trees** and **Graphs**
2	Recurrence Relations	[*Rigorous*] Formulation and Solution Techniques
3	Data Structures	[*In-depth and breadth -typical end-to-end coverage of a* ***Data Structures & Algorithms course***]: linearly ordered and partially ordered **data and data structures**, **graphs** – representation and basic algorithms; **Sorting** algorithms;
4	Divide-and-Conquer	[*Basic*]
5	Order Complexity	[*Basic*] Notation (big-O and big-Omega and theta) [*Rigorous*] Growth rate, and Asymptotic complexity;
6	Logic	[*Rigorous*]Boolean algebra / Propositional Logic: **Satisfiability** Problem
7	Theory of Computation	[*Basic*] Turing Machines and Computability; Context Free Grammars

3.b. Lecture Schedule:

Lec. #	Topic	Learning Outcome(s) [***The student should be able to***:]	Reading
L1.0	Course Introduction	-	
L1.1	Machine Model: RAM, Uniform Cost Model vs. Logarithmic Cost Model;	• statesalient features of the RAM model and its implications for designing / analyzing algorithms	-
L2.1	Design:Review of Divide-and-Conquer: Basic Pattern and Example	• apply divide-and-conquer to design algorithms for simple problems	T1 Sec.4.1
L2.2	Design:Divide-and-Conquer: (Structural) Induction:Examples and Typical Cases	• use structural induction on different forms of data (or data structures) for designing algorithms	-
L2.3	Analysis: Locality of Reference	• explain how locality (or the lack of it) may impact the performance of an algorithm	-

L3.1	Design:Divide-and-Conquer – Example: Matrix Multiplication – Classic algorithm and its complexity	• analyze the cost of multiplying two matrices using the classic algorithm or its variants	T1 Sec. 4.2
L3.2	Design:Divide-and-Conquer – Example: Matrix Multiplication – Blocking and its implication, Strassen's algorithm and its complexity	• design divide-and-conquer algorithms on matrices using blocking • explain the efficiency achieved by Strassen's algorithm	T1 Sec. 4.2
L4.1	Design:Issues in Top-Down Design: Overlapping Sub-problems	• identify overlapping sub-problems, if they exist, in any top down design • calculate the impact of overlapping sub-problems on the cost of the solution	T1 Sec.15.1
L4.2	Design:Top-Down Design: Memoization – Example.	• memoize the results in a given top-down-design solution and • analyze the impact of memoization on the cost of the solution	T1 Sec.15.1 and Sec. 15.3
L5.1	Design:Top-down Design vs. Bottom-up Design	• state the pros and cons of top-down design and bottom-up-design • choose between top-down-design and bottom-up-design as a design approach to a given problem	-
L5.2	Design: Dynamic Programming – Function Problems – Examples	• apply dynamic programming to solve function problems	-
L6.1	Design: Top-Down Design: Optimal Sub-structure Property – Example.	• discover and argue that an optimization problem exhibits optimal substructure • formulate a recurrence for an optimization problem	T1 Sec.15.2 – 15.3
L6.2	Design: Dynamic Programming – Optimization Problems – Motivating Example 1	• formulate a recurrence relation based on optimal sub-structure; • write a dynamic programming (DP) algorithm based on the recurrence; • optimize the space requirement for the DP algorithm	T1 Sec.15.3-15.5

Birla Institute of Technology & Science, Pilani
Pilani Campus, VidyaVihar
Pilani 333031, Rajasthan, India
Tel: +91 1596 245073
Fax: +91 1596 244183
Web: www.pilani.bits-pilani.ac.in

L7.1	Design: Dynamic Programming – Optimization Problems – Motivating Example 2	- ditto -	T1 Sec.15.3-15.5
L7.2	Design: Dynamic Programming - Problem Domain: Text/String Processing Problems: Application: DNA Sequence Alignment	• formulate and solve typical text/string processing problems using DP	AR
L7.3	Design: Dynamic Programming - Problem Domain: Text/String Processing Problems: Approximate String Matching and Text Matching – Application(s)	• recognize variants of string matching / text matching problems • adapt DP solution for a given variant	AR, T1 Sec. 15.4
L8.1	Design: Dynamic Programming – Problem Domain: Graph Problems: All Pairs Shortest Paths	• formulate and solve typical graph problems using DP	T1 Sec. 25.2
L8.2	Design: Dynamic Programming – Problem Domain: Graph Problems: Transitive Closure	• recognize variants of a graph problem • adapt DP solution for a given variant	-
L8.3	Design: Dynamic Programming – Problem Domain: Graph Problems: Application: Translating Finite Automata to Regular Expressions	- ditto -	-
L9.1	Design: Greedy Algorithms: Motivating Examples	• recognize problems where Greedy technique would be applicable • apply Greedy Technique to solve problems	T1 Sec. 16.1
L9.2	Design: Greedy Choice Property and Optimal Substructure Property; Limitation of Greedy Technique.	• verify whether Greedy Choice applies to a given problem • distinguish between Optimal Substructure and Greedy Choice for a given problem • recognize problems where Greedy technique would not be applicable	T1 Sec. 16.2-16.3
L10.1	Design:Greedy Technique: Graph Problems - Example: Algorithms for Minimum Spanning Tree	• apply Greedy Technique to solve Graph problems	T1 Ch 23
L10.2	Design:Greedy Technique: Counter-Example: Steiner Tree problem	• recognizeproblems where Greedy technique would not be applicable	-

Birla Institute of Technology & Science, Pilani
Pilani Campus, VidyaVihar
Pilani 333031, Rajasthan, India
Tel: +91 1596 245073
Fax: +91 1596 244183
Web: www.pilani.bits-pilani.ac.in

L11.1	Design: Theory of Greedy Algorithms:Matroids: Introduction and Examples.	• formulate a given problem using matroids	T1 Sec. 16.4
L11.2	Design: Theory of Greedy Algorithms:Weighted Matroids – Definition and Examples.	• formulate a given problem using weighted matroids	T1 Sec. 16.4
L11.3	Design:Theory of Greedy Algorithms: Weighted Matroids and Greedy Algorithms	• state the relation between weigtedmatroids and greey algorithms	T1 Sec. 16.4
L11.4	Design:Theory of Greedy Algorithms: Template Greedy Algorithm	• formulate a given problem using matroids and instantiate a greedy algorithm for the specific problem	T1 Sec. 16.4 – 16.5
L12.1	Problems: Online Problems and Algorithms: Example Problem and Typical Algorithms	• distinguish between online and offline problems • distinguish between online algorithms and offline algorithms	AR
L12.2	Analysis: Analysis of Online Algorithms: Worst-case analysis vs. Competitiveness Analysis	• distinguish between worst-case analysis of algorithms and competitiveness analysis of algorithms • perform competitiveness analysis of algorithms for online problems	AR
L13	Analysis: Amortized Analysis: Introduction and Examples	• perform amortized analysis of algorithms for online problems	AR
L14.1	Design: Randomization: Motivation for Randomized Algorithms and Example (QuickSort)	• identify scenarios where randomization would be applicable (in designing an algorithm)	T1 Sec. 7.3
L14.2	Analysis: Analysis of Randomized Algorithms – Example.	• differentiate analysis of randomized algorithms from that of deterministic algorithms	T1 Sec. 7.3
L14.3	Design: Randomized Algorithms : Models : Las Vegas vs. Monte Carlo	• distinguish between Las-Vegas and Monte-Carlo algorithms	-
L15.1	Design: Game Tree Evaluation: Context, Definition, and a Deterministic Algorithm.	• -	AR
L15.2	Design: Randomized Algorithms : Las Vegas Algorithms: Example (Game Tree Evaluation)	• design a Las Vegas algorithm for a given problem • analyze a Las Vegas algorithm	AR

Birla Institute of Technology & Science, Pilani
Pilani Campus, VidyaVihar
Pilani 333031, Rajasthan, India
Tel: +91 1596 245073
Fax: +91 1596 244183
Web: www.pilani.bits-pilani.ac.in

L16.1	Problem Domain: Number Theory: Computing GCD and Euclid's Algorithm	• state the correctness and complexity of Euclid's algorithm	T1 Sec. 31.2
L16.2	Problem Domain: Number Theory: Aryabhatia's algorithm for computing GCD and Linear coefficients	• state the correctness and complexity of Aryabhatia's algorithm	T1 Sec. 31.2
L16.3	Problem Domain: Number Theory: Groups and Modular Arithmetic; Groups Z_n and Z_n^*, Size of Z_n^* and computing the size of Z_n^*	• formulate modular arithmetic problems as groups • design an algorithm to compute the size of Z_n^* • relate the complexity of factoring an integer and that of computing the size of Z_n^*	T1 Sec. 31.3
L17.1	Problem Domain: Number Theory: Properties of Z_n^*; Euler's Theorem and Fermat's Theorem	• relate properties of Z_n^* to Euler's Theorem. • relate Euler's Theorem to Fermat's Theorem • state the issue in using Fermat's Theorem in distinguishing between prime numbers and composite numbers	T1 Sec. 31.3
L17.2	Problem Domain: Cryptography: Secrecy and Encryption Systems: Public Key Cryptography	• state the secrecy or confidentiality requirement as a computational problem • distinguish between Shared Key Encryption and Public Key Encryption • state the computational requirements and the pros and cons of using Public Key Key Encryption	T1 Sec. 31.7
L17.3	Problem Domain: Cryptography: RSA Encryption Protocol: Design and Correctness	• argue the communication correctness of RSA	T1 Sec. 31.7
L18.1	Problem Domain: Cryptography: RSA Encryption Protocol: Efficiency and Security	• argue the computational requirements for sender and receiver in using RSA • provide a rigorous – even if informal – argument of the security of RSA	T1 Sec. 31.7

Birla Institute of Technology & Science, Pilani
Pilani Campus, VidyaVihar
Pilani 333031, Rajasthan, India

Tel: +91 1596 245073
Fax: +91 1596 244183
Web: www.pilani.bits-pilani.ac.in

L18.2	Problem Domain: Number Theory: Primality Testing: Motivation, Cost of typical deterministic algorithms	• define the problem of testing for primes • derive the complexity of typical deterministic algorithms for testing primality	
L18.3	Problem Domain: Number Theory: Primality Testing: Cost of Finding a prime number	• compute the expected cost of finding a prime number assuming the cost of testing for primes	T1 Sec. 31.8
L18.4	Problem Domain: Number Theory: Primality Testing: A pseudo-primality testing algorithm	• explain the design and limitations of a primality testing algorithm based on Fermat's Theorem	T1 Sec. 31.8
L19	Problem Domain: Number Theory: Primality Testing: Miller-Rabin Test: Algorithm, Time Complexity, and Error Bounds.	• explain the Miller-Rabin Test • argue the time complexity and error bounds of the Miller-Rabin Test	T1 Sec. 31.8
L20.1	Complexity: Lower Bound Analysis of Problems – Examples (Searching and Sorting).	• provide intuitive explanations of lower bounds for searching and sorting problems	R1 Sec. 10.1
L20.2	Complexity: Lower Bound Analysis of Problems: Decision Tree method	• apply the Decision Tree method for analyzing problems for lower bounds	R1 Sec. 10.1
L20.3	Complexity: Reduction: Models: Karp Reduction and 1-1 Reduction – Examples	• explain the notion of reduction and its relation to complexity of problems • apply reductions for specific problems • distinguish between the use of Karp reduction and 1-1 reduction	-
L21.1	Complexity: Lower Bound Analysis of Problems: Reduction method	• apply reductions for analyzing problems for lower bounds	R1 Sec. 10.3
L21.2	Complexity: Reduction: Models: Turing Reduction and the relation between different reductions	• explain Turing reduction and its use • apply Turing reduction on problems • relate other reductions to Turing reduction	-

L22.1	Complexity: Computability Review: Turing Machine Model, Church-Turing Hypothesis, Equivalence of TM and RAM models	• explain Church-Turing hypothesis and its significance • argue equivalence between Turing Machines and Random Access Machines	-
L22.2	Complexity: Computability Review: Non-computable Functions – Examples and Cardinality	• provide examples of non-computable functions • argue that a given function is not computable by using reduction • argue that the set of non-computable functions form an uncountable set	-
L23.1	Complexity: Complexity Classes: Time Complexity Classes and Space Complexity Classes	• prove membership of a given problem in a specific complexity class • argue the relation between time complexity classes and space complexity classes	T1 Sec. 34.1
L23.2	Complexity: Tractability: Polynomial Time vs. Exponential Time, Classes P and EXP.	• provide examples of problems in P and EXP • prove membership of problems in P	T1 Sec. 34.1
L24.1	Complexity: Non-deterministic Computation: Ideas and Examples	• write non-deterministic algorithms for given problems	T1 Sec. 34.2
L24.2	Complexity: Non-deterministic Computation: Certificate Verification model	• explain non-deterministic algorithms using the Certificate Verification model	T1 Sec. 34.2
L24.3	Complexity: Non-deterministic Computation: Non-deterministic Time Complexity Classes, Class NP, Example Problems in NP.	• relate deterministic time complexity classes to non-deterministic time complexity classes • prove membership of problems in NP	T1 Sec. 34.2
L25.1	Complexity: P vs. NP	• explain why P is a subset of NP • explain the implication of P being a proper subset of NP • explain the implication of NP being a subset of P	-

L25.2	Complexity: NP-Completeness	• explain the notion of NP-completeness • explain the implication of a problem being NP-complete	T1 Sec. 34.3
L25.3	Complexity: NP-Completeness of Circuit-SAT	• explain intuitively why Circuit-SAT is NP-complete	T1 Sec. 34.3
L26.1	Complexity: Transitivity of Reductions; Proving NP-completeness via reduction	• explain why reductions are transitive and how transitivity helps in proving NP-completeness of a problem	T1 Sec. 34.3
L26.2	Complexity: NP-completeness via reduction: Examples (SAT and its variants)	• prove that SAT and some of its variants and special cases are NP-complete	T1 Sec. 34.3-34.5
L27	Complexity: NP-completeness via reduction: Techniques for Reduction - examples	• apply different techniques to reduce a NP-hard problem to another	T1 Sec. 34.4
L28.1	Complexity: The class NPC	• state the nature and properties of the class NPC	T1 Sec. 34.4-34.5
L28.2	Complexity: NP-completeness: Structure of Problems – What makes problems hard?	• explain intuitively why apparently simple variants of problems in P are NP-complete	-
L29.1	Complexity: Complexity Classes: NP and coNP: Complements of Problems, Closures of Complexity Classes	• explain the relation and difference between classes NP and coNP	T1 Sec. 34.2, AR
L29.2	Complexity: Complexity Classes: coNP, Problems in coNP, Intersection of NP and coNP	• prove that a given problem is in coNP • explain the implication of membership in the intersection of NP and coNP • explain the relation between P vs. NP and NP vs. co-NP	T1 Sec. 34.2, AR
L30.1	Design: Dealing with Hard problems	• characterize outcomes / trade-offs of a given approach to deal with a hard problem	

Birla Institute of Technology & Science, Pilani
Pilani Campus, VidyaVihar
Pilani 333031, Rajasthan, India

Tel: +91 1596 245073
Fax: +91 1596 244183
Web: www.pilani.bits-pilani.ac.in

		• decide and/orchoose an approach for dealing with a hard problem	
L30.2	Design: Dealing with Hard problems: Exact Solutions: Backtracking: Motivating Example and Solution	• explain how search works in general and the role of backtracking in search	-
L31.1	Design: Dealing with Hard problems: Exact Solutions: Backtracking: Template solution and example solution	• explain how a generic backtracking template works • design algorithms for specific problems using the backtracking template	R1 Ch. 7
L31.2	Design: Dealing with Hard problems: Exact Solutions: Backtracking: Application Domain	• solve problems using search with backtracking as a generic strategy	R1 Ch. 7
L32	Design: Dealing with Hard problems: Exact Solutions: Branch-and-Bound	• relate Branch-and-Bound to Backtracking • design algorithms for specific problems using branch-and-bound	R1 Ch. 8
L33	Design: Dealing with Hard problems: Exact Solutions: Branch-and-Bound: Use of Heuristics	• explain and illustrate the role of heuristics in solving specific problems using branch-and-bound • design heuristics to improve the performance of branch-and-bound algorithms in solving specific problems	R1 Ch. 8
L34.1	Problems: Optimization Problems: Examples	• formulate optimization problems crisply	-
L34.2	Complexity: Complexity Classes: PO and NPO, Example problems in PO and NPO.	• decide membership of specific problems in PO and NPO	AR

Birla Institute of Technology & Science, Pilani
Pilani Campus, VidyaVihar
Pilani 333031, Rajasthan, India

Tel: +91 1596 245073
Fax: +91 1596 244183
Web: www.pilani.bits-pilani.ac.in

L34.3	Complexity: Complexity Classes: NP-hard Optimization Problems – Examples.	• prove NP-hardness of optimization problems using reductions	AR
L35.1	Design: Dealing with Hard problems: Approximation: Introduction to Approximation Algorithms – Vertex Cover as an Example.	• explain the notion of approximating the solution and that of approximation algorithms	T1 Sec. 35.1
L35.2	Design: Dealing with Hard problems: Approximation: Lower-Bounding as a technique for Designing Approximation Algorithms - Example	• explain the significance of lower-bounding in designing approximation algorithms	AR T1 Sec. 35.1
L35.3	Design: Dealing with Hard problems: Approximation Algorithms: Absolute Error and Absolute Approximation; Example and Counter-example.	• assess – informally – whether absolute approximation is feasible for a given problem • design simple absolute error approximation algorithms for specific problems • explain how absolute approximation is an inadequate approach • prove that certain problems are not approximable with an absolute error	AR
L36.1	Design: Dealing with Hard problems: Approximation Algorithms: Relative Error and Approximation within Constant Factor - Example	• explain how to bound the error of an approximation algorithm by a constant factor	AR
L36.2	Design: Dealing with Hard problems: Approximation Algorithms – Design Techniques: Greedy Technique	• design approximation algorithms using the greedy technique	AR T1 Sec. 35.3
L36.3	Design: Dealing with Hard problems: Approximation Algorithms – Design Techniques: Sequencing	• design approximation algorithms using sequencing	AR
L37.1	Design: Dealing with Hard problems: Approximation Algorithms – Design Techniques: Randomization	• design approximation algorithms using randomization	T1 Sec. 35.4

Birla Institute of Technology & Science, Pilani
Pilani Campus, VidyaVihar
Pilani 333031, Rajasthan, India
Tel: +91 1596 245073
Fax: +91 1596 244183
Web: www.pilani.bits-pilani.ac.in

L37.2	Design: Dealing with Hard problems Approximation Algorithms – - Problem Domain: Graph Problems: Steiner Tree Problem	• design approximation algorithms for specific graph problems	AR
L38.1	Design: Dealing with Hard problems Approximation Algorithms – - Problem Domain: Graph Problems: Metric TSP	• design approximation algorithms for specific graph problems	T1 Sec. 35.2
L38.2	Complexity: Complexity Classes: Class APX – Examples	• prove that specific problems are in APX	AR
L38.3	Complexity: Non-approximability: TSP is not r-approximable.	• prove that specific problems are not r-approximable	T1 Sec. 35.2.2
L39	Design: Dealing with Hard problems: Approximation Algorithms – Polynomial Time Approximation Schemes	• design PTAS for NP-hard optimization problems	AR
L40	Summary and Conclusion	• state what you have learnt in this course and where you can apply what you have learnt • state what you can learn further on your own based on what you have learnt	-

4. Evaluation

4. a. Evaluation Model

Component	Nature of Assessment	Intent	Form / Format
Quizzes (3)	• Design / Analysis of algorithm(s) for a specific problem • Application of basic concepts	Review of immediate material; Focus on one or a few topics	In class, Written, Timed, Open Book.

Assignment – Phase I	Reading, Understanding Writing, and Explaining Research	Self Learning	Take Home; Teams of 2 or 3; Individual contribution will be assessed as well.
Assignment – Phase II	Adapting and Applying Research	Self Learning	- ditto -
Mid-Term Test	• Design / Analysis of algorithm(s) for a specific problem • Application of techniques in formulating problems, designing and analyzing problems. • Analyzing problems, Performing reductions.	Sound and extensive understanding of topics taught in class; Ability to apply and relate what is taught in class to problems not known apriori.	Scheduled centrally; Written, Timed, Open Book.
Comprehensive Exam	• Design / Analysis of algorithm(s) for a specific problem • Application of techniques in formulating problems, designing and analyzing problems. • Analyzing problems, Performing reductions. • Analyzing and proving complexity of problems and relations between complexity classes • Approaching hard problems	Sound and comprehensive understanding of topics taught in class; Ability to apply and relate what is taught in class to problems in the external world. Ability to apply algorithmic thinkingfor understanding and solving problems.	Scheduled centrally; Written, Timed, Open Book.

4. b. Evaluation Scheme:

Component	Weight	Date& Time	Duration	Remarks
Quizzes (3)	3x15M= 45M	In Jan., Feb. and Apr.	30 to 45 min. each	**Actual date will be announced one lecture prior to the quiz (i.e. one or two days notice).** *Read make-up policy below.*

Assignment – Phase I	20M	In Feb.	10 to 15 days	Actual date will be announced one day before the start. *Read late submission policy below*.
Assignment – Phase II	35M	In Mar. – Apr.	15 to 25 days	Actual date will be announced one day before the start. *Read late submission policy below.*
Mid-Term Test	33M	16/3 9:00 - 10:30 AM	90 minutes	*Read Make-up Policy below.*
Comprehensive Exam	67M	7/5 FN	180 minutes	*Read Make-up Policy below.*
TOTAL	200M			

4. c. Make-up Policy:

- **Quizzes:**
    1. **A student may avail at most one make-up for all three quizzes** put together.
    2. The make-up quiz will be scheduled after all the regular quizzes.
    3. Coverage for the make-up quiz will be that of all three quizzes put together.
- **Quizzes and Mid-Term:**
    1. Make-up for quizzes or for the mid-term test will be ***granted only for genuine reasons*** when the student is physically unable to appear for the quiz/test.
    2. It is the responsibility of the student to communicate a make-up request to one of the instructor(s) before or during the test/quiz.
    3. Decision of the instructor-in-charge with respect to 1. and 2. above is final.
- **Assignment:**
    1. **Late submission of assignment will incur a penalty** of **25% up to 24 hours and a penalty of 50% up to 48 hours past the deadline** for that phase.
    2. No further submissions will be entertained 48 hours past the deadline.
- **Comprehensive Exam:**
    1. Permission for a Make-up for the comprehensive exam will have to be obtained from **Dean Instruction** and
    2. **Make-up for the comprehensive exam will usually be scheduled centrally**.

4.d. Fairness Policy:

- Fairness policies of BITS Pilani, as stated in Academic Regulations and Guidelines to Students, are all applicable for this course.
- In addition, the following are applicable in the context of take-home assignment for this course:
    - Teams are expected to work on their own on assignments.

Birla Institute of Technology & Science, Pilani
Pilani Campus, VidyaVihar
Pilani 333031, Rajasthan, India
Tel: +91 1596 245073
Fax: +91 1596 244183
Web: www.pilani.bits-pilani.ac.in

- Students are allowed to consult/discuss with other students/teams for the take-home assignment but such consultation/discussion should be explicitly acknowledged and reported to the instructor prior to evaluation.
- All students are expected to contribute equally within a team:
  - **Individual contributions should be identified and documented in qualitative and quantitative terms by the team members.**
  - The instructor's assessment regarding the contributions of team members would be final.
- All material referred to should be explicitly included in Bibliography / References of the reports submitted as part of the assignment.
- Contents of reports should not be included as is from other sources.

- Unfair means would include:
  - copying from other students or enabling copying by other students;
  - copying / borrowing material from the Web or from other sources of information including all electronic sources.
- Any use of unfair means in quizzes, assignment, or test/exam will be handled strictly:
  - The minimum penalty would be loss of full weight of the component.
  - In addition, students involved in such activity are liable for further sanctions including being formally reported to the Unfair Means committee of the Department as well as to the Examination Committee in which case they will be subject to penalties enabled by Unfair Means Rules of the Institute.

5. Consultation Hours: TuTh 2.30pm to 3.30pm (Room 6120-L, CSIS Dept., NAB).

6. Notices:<u>**All notices**</u> concerning this course will be displayed on the <u>**course website (on Nalanda) only**</u>. If there is a need <u>*email would be used on short notice (12 hours)*</u> – only BITS Pilani mail id of students would be used.

Instructor -In- Charge
CS F364

Birla Institute of Technology & Science, Pilani
Pilani Campus, VidyaVihar
Pilani 333031, Rajasthan, India

Tel: +91 1596 245073
Fax: +91 1596 244183
Web: www.pilani.bits-pilani.ac.in