NEW YORK CITY COLLEGE OF TECHNOLOGY
The City University of New York

DEPARTMENT:	Mathematics
COURSE:	MAT 2540
TITLE:	Discrete Structures and Algorithms II
DESCRIPTION:	This course continues the discussion of discrete mathematical structures and algorithms introduced in MAT 2440. Topics in the second course include predicate logic, recurrence relations, graphs, trees, digital logic, computational complexity and elementary computability.
TEXT:	Discrete Mathematics and its Applications,7th edition by Kenneth H. Rosen McGraw-Hill
CREDITS:	3 (2 class hours, 2 lab hours)
PREREQUISITES:	MAT 2440

Prepared by Professors Henry Africk, Brad Isaacson, Caner Koca, Nan Li, Satyanand Singh, Arnavaz Taraporevala, Johann Thiel (Fall 2017)

A. Testing Guidelines:
   The following exams should be scheduled:
   1. A one-hour exam at the end of the First Quarter
   2. A one-session exam at the end of the Second Quarter
   3. A one-hour exam at the end of the Third Quarter
   4. A one-session Final Examination

B. A Computer Algebra System will be used in class and for a project.

Course Intended Learning Outcomes/Assessment Methods

Learning Outcomes	Assessment Methods
1. Understand and apply the concept of mathematical induction.	Classroom activities and discussion, homework, project, exams.
2. Identify and solve counting problems.	Classroom activities and discussion, homework, project, exams.
3. Compare data structures and algorithms.	Classroom activities and discussion, homework, project, exams.
4. Use the master theorem to solve recurrences that arise from divide-and-conquer algorithms.	Classroom activities and discussion, homework, project, exams.
5. Study graphs and trees, and solve algorithmic problems such as finding shortest paths or spanning trees.	Classroom activities and discussion, homework, project, exams.

General Education Learning Outcomes/Assessment Methods

Learning Outcomes	Assessment Methods
1. Gather, interpret, evaluate, and apply information discerningly from a variety of sources.	Classroom activities and discussion, homework, project, exams.
2. Understand and employ both quantitative and qualitative analysis to solve problems.	Classroom activities and discussion, homework, project, exams.
3. Employ scientific reasoning and logical thinking.	Classroom activities and discussion, homework, project, exams.
4. Communicate effectively using written and oral means.	Classroom activities and discussion, homework, project, exams.
5. Utilize computer based technology in accessing information, solving problems and communicating.	Classroom activities and discussion, homework, project, exams.
6. Work with teams. Build consensus and use creativity.	Classroom activities and discussion, project, homework.
7. Acquire tools for lifelong learning.	Classroom activities and discussion, homework, project, exams.

New York City College of Technology Policy on Academic Integrity

Students and all others who work with information, ideas, texts, images, music, inventions, and other intellectual property owe their audience and sources accuracy and honesty in using, crediting, and citing sources. As a community of intellectual and professional workers, the College recognizes its responsibility for providing instruction in information literacy and academic integrity, offering models of good practice, and responding vigilantly and appropriately to infractions of academic integrity. Accordingly, academic dishonesty is prohibited in The City University of New York and at New York City College of Technology and is punishable by penalties, including failing grades, suspension, and expulsion. The complete text of the College policy on Academic Integrity may be found in the catalog.

MAT 2540 Discrete Structures and Algorithms II Text: Discrete Mathematics and its Applications,7th edition, by Rosen

Lec.	Discrete Structures and Algorithms II	Homework
1	5.2 Strong Induction and Well-Ordering (333-341)	(P. 341) 3-7
2	5.3 Recursive Definitions and Structural Induction (344-347*)	(P. 357) 1-9 odd, 20*
3	5.4 Recursive Algorithms (360-370)	(P. 370) 1, 3, 5, 7-11, 29-31, 44-48
4	6.3 Permutations and Combinations (407-413) (optional)	((P. 413) 1-7, 9, 11, 17, 19, 31
5	6.1 The Basics of Counting (Skip trees) (385-394)	(P. 396) 1-15 odd, 19, 21, 25, 29, 33, 37, 45, 47, 49
6	6.2 The Pigeonhole Principle (399-404)	(P. 405) 1-9 odd, 15, 19
7	**Review**	
8	**Test 1**	
9	8.1 Applications of Recurrence Relations (501-505)	(P. 510) 1, 3, 4, 7-9, 11, 13, 21, 26, 27, 28
10-11	8.2 Solving Linear Recurrence Relations (514-524)	(P. 524) 1, 3, 7, 9, 11, 23-25, 27-30
12-13	8.3 Divide-and-Conquer Algorithms and Recurrence Relations (527-534)	(P. 535) 1, 7, 9-17, 21, 36-37
14	10.1 Graphs and Graph Models (641-649)	(P. 649) 3-9, 27-29, 33
15	10.2 Graph Terminology and Special Types of Graphs (651-665)	(P. 665) 1-10, 20, 21-25, 35, 36-43, 59, 61
16	10.3 Representing Graphs and Graph Isomorphism (668-675)	(P. 675) 1-23 odd, 26, 27, 30, 31, 35-41 odd
17	**Review**	
18	**Test 2**	
19	10.4 Connectivity (skip counting paths) (678-689)	(P. 689) 1-5, 11, 31-34, 50, 54
20	10.5 Euler and Hamilton Paths (693-703)	(P. 703) 1-7 odd, 9, 10, 13-15, 19, 21, 30-36, 37-39, 47
21	10.6 Shortest-Path Problems (707-716)	(P. 716) 2-7, 11, 17. 25, 27
22	11.1 Introduction to Trees (745-755)	(P. 755) 1-9 odd, 17-20, 27, 28
23	11.2 Application of Trees (757-769)	(P. 769) 1-4, 6-8, 11, 19-22, 37
24	11.3 Tree Traversal (772-782)	(P. 783) 1, 3, 6, 7-15, 22-24
25	**Review**	
26	**Test 3**	
27	11.4 Spanning Trees 11.5 Minimum Spanning Trees	(P. 795) 2-6, 13-18, 27, 28, 30 (P. 802) 1-3, 6, 7
28	5.3 Recursive Definitions and Structural Induction (Recursively Defined Sets and Functions) (349-356)	(P. 358) 22, 23, 25, 32, 33, 34-36, 43, 44
29	**Review**	
30	**Final Exam**	

(see the next page for a list of suggested projects)

List of Suggested Projects

1. Lecture 1: Ackermann's Function (Section 5.3, P. 359, Exercise 48-55)
2. Lecture 2: QuickSort Algorithm (Section 5.4, P. 371, Exercise 49-52)
3. Lecture 9: A variation to the Towers of Hanoi Game (Section 8.1, P. 512, Exercise 32)
4. Lecture 9: The Josephus Problem (Section 8.1, P. 512, Exercise 33-37)
5. Lecture 11: Lucas Numbers (Section 8.2, P. 525, Exercise 11)
6. Lecture 15: Coding Graphs (nodes and edges) in a programming language (e.g. Python, C++, or Java)
7. Lecture 22: Coding Family Trees, and Boolean functions checking relationships between family members
8. Lecture 23: Coding Binary Search Trees (e.g. in Python, C++, or Java)
9. Lecture 28: Coding a recursive algorithm that computes the height of any tree; for example, a binary search tree.