OFFICIAL UNDERGRADUATE COURSE OUTLINE (page 1)

COURSE IMPLEMENTATION DATE:	May 1994
COURSE REVISED IMPLEMENTATION DATE:	January 2013
COURSE TO BE REVIEWED:	November 2018
(six years after UEC approval)	*(month, year)*

OFFICIAL UNDERGRADUATE COURSE OUTLINE INFORMATION

Students are advised to keep course outlines in personal files for future use.
Shaded headings are subject to change at the discretion of the department – see course syllabus available from instructor

MATH 343	SCIENCE/MATH	3
COURSE NAME/NUMBER	FACULTY/DEPARTMENT	UFV CREDITS

APPLIED DISCRETE MATHEMATICS
COURSE DESCRIPTIVE TITLE

CALENDAR DESCRIPTION:

Just as there is a continuum between applied math and theoretical physics, there is a continuum between applied discrete math and theoretical computing. This course is an introduction to algorithms, but with a discrete math – rather than a computing – emphasis. In particular, this course will cover some standard algorithms in combinatorics, running time analysis, correctness of algorithms, and techniques for selecting an appropriate algorithm to solve a problem.

PREREQUISITES: One of MATH 225, MATH 221, or COMP 251
COREQUISITES:
PRE or COREQUISITES:

SYNONYMOUS COURSE(S):
(a) Replaces:
(b) Cross-listed with:
(c) Cannot take: for further credit.

SERVICE COURSE TO: *(department/program)*

TOTAL HOURS PER TERM: 60

STRUCTURE OF HOURS:

Lectures:	**60**	Hrs
Seminar:		Hrs
Laboratory:		Hrs
Field experience:		Hrs
Student directed learning:		Hrs
Other (specify):		Hrs

TRAINING DAY-BASED INSTRUCTION:
Length of course:
Hours per day:

OTHER:
Maximum enrolment: **36**
Expected frequency of course offerings: Every second year
(every semester, annually, every other year, etc.)

WILL TRANSFER CREDIT BE REQUESTED? (lower-level courses only)	☐ Yes	☐ No
WILL TRANSFER CREDIT BE REQUESTED? (upper-level requested by department)	☐ Yes	☒ No
TRANSFER CREDIT EXISTS IN BCCAT TRANSFER GUIDE:	☐ Yes	☒ No

Course designer(s):	**Cynthia Loten, Joseph Yu, Ian Affleck**		
Department Head:	**Cynthia Loten**	Date approved:	**August 29, 2012**
Campus-Wide Consultation (CWC)		Date of meeting:	**September 29, 2012**
Curriculum Committee chair:	**David Fenske**	Date approved:	**October 19, 2012**
Dean/Associate VP:	**Lucila Lee**	Date approved:	**October 19, 2012**
Undergraduate Education Committee (UEC) approval		Date of meeting:	**November 23, 2012**

LEARNING OUTCOMES:

Upon successful completion of this course, students will be able to:

1. Implement well known algorithms in combinatorics
2. Analyse the average case and worse case complexity of an algorithm.
3. Model a problem and use an appropriate algorithm to solve the problem.
4. Demonstrate the correctness of an algorithm.

METHODS: *(Guest lecturers, presentations, online instruction, field trips, etc.)*

Lectures

METHODS OF OBTAINING PRIOR LEARNING ASSESSMENT RECOGNITION (PLAR):

☐ Examination(s) ☐ Portfolio assessment ☐ Interview(s)

☒ Other (specify): Please check the university calendar for the course challenge policy.

☐ PLAR cannot be awarded for this course for the following reason(s):

TEXTBOOKS, REFERENCES, MATERIALS:

[Textbook selection varies by instructor. An example of texts for this course might be:]

The textbook is chosen by a departmental curriculum committee. Recommended texts are:
J. Kleinberg, E. Tardos. 2005. *Algorithm Design*
A. Levitin. 2011. *The Design and Analysis of Algorithms*

SUPPLIES / MATERIALS:

STUDENT EVALUATION:

[An example of student evaluation for this course might be:]

The weighting of the various components may vary from instructor to instructor and from year to year, although there must be at least two midterms, and the comprehensive final exam must be worth from 30% to 50% of the final grade. A student must obtain at least 40% on the final exam in order to pass this course.

Quizzes	10%
Assignments	10%
Tests (2)	40%
Final exam	40%

COURSE CONTENT:

[Course content varies by instructor. An example of course content might be:]

1. Concepts of combinatorics and graph theory: combinations, permutations, partitions, trees, paths and cycles
2. Computer representation of combinatorial objects
3. Sorting, searching, string matching and min/max algorithms
4. Running time analysis of algorithms: worst-case and average-case analysis, asymptotic orders of growth
5. Running time complexity classes: Polynomial (P), Non-Deterministic Polynomial (NP) , NP-complete (NP-c) and NP-hard
6. Heuristics and approximation algorithms
7. Bin packing, vertex cover and graph colouring algorithms
8. Greedy Algorithms
9. Randomized Algorithms