Knight Foundation School of Computing and Information Sciences

Course Title: Computational Thinking **Date:** 10/16/2023

Course Number: COP 3XXX

Number of Credits: 3

Subject Area: Programming	**Subject Area Coordinator:** Janki Bhimani **email:** jbhimani@fiu.edu
Catalog Description: Computational thinking principles, covering algorithms, data structures, problem-solving, problem decomposition, creativity, and topics in recursion and ethical considerations in computing.	
Textbooks: Applied Computational Thinking with Python: Design algorithmic solutions for complex and challenging real-world problems by Sofía De Jesús and Dayrene Martinez. ISBN-13: 978-1839219436. Publisher: Packt Publishing. Date: November 27, 2020	
References (for further reading): Introduction to Algorithms: A Comprehensive Guide for Beginners: Unlocking Computational Thinking by Cuantum Technologies. ISBN-13: 979-8854326957. Independently published. Date: July 30, 2023	
Prerequisites Courses: COP 2XXX - Python Programming I or COP 2210 - Programming I or COP 2250 - Programming in Java or Advisor's Permission	
Corequisite Courses: COP 2XXX - Introduction to Data Science and COP 3XXX - Python Programming II	

Type: Core Course for BS in Data Science; Elective for CS and IT Majors.

Prerequisite Topics:

1. Programming fundamentals such as control structures, basic data types and structures, functions, and object-oriented paradigm in at least one programming language.
2. Experience in solving simple computational problems using coding.

Course Outcomes:

1. **Comprehend** the key principles of computational thinking, including abstraction, decomposition, pattern recognition, and algorithmic design.
2. **Develop** and implement efficient algorithms for problem-solving.
3. **Explain** and apply propositional logic in computer science contexts, including the syntax, semantics, and truth tables.
4. **Critically evaluate** arguments and reasoning through inference rules, recognizing and avoiding logical fallacies.
5. **Break down** complex problems into manageable tasks or subproblems using top-down design and stepwise refinement.
6. **Describe** the essence of computational creativity and its relevance in modern computing.

7. **Design** solutions that integrate creative algorithms, considering elements like randomness and generative art.
8. **Implement** recursive functions, understanding the differences and trade-offs between recursion and iteration.
9. **Critically assess** computing solutions, considering ethical implications like privacy, security, AI biases, and intellectual property rights.

Association between Student Outcomes and Course Outcomes

BS in Computing: Student Outcomes	Course Outcomes
1) Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.	1,2,5,7,8
2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.	1,2,5,7,8
3) Communicate effectively in a variety of professional contexts.	6,9
4) Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.	9
5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.	

Program Specific Student Outcomes	
6) Apply theory, techniques, and tools throughout the data science lifecycle and employ the resulting knowledge to satisfy stakeholders' needs. [DS]	2,3,7,9

Assessment Plan for the Course and how Data in the Course are used to assess Student Outcomes

Student and Instructor Course Outcome Surveys are administered at the conclusion of each offering, and are evaluated as described in the School's Assessment Plan: https://abet.cis.fiu.edu/

Knight Foundation School of Computing and Information Sciences
COP 3XXX Computational Thinking

Outline

Topic	**Number of Lecture Hours** (Total: 37.5 hours = 15 weeks * 2 lectures/week * 1.25 hrs/lecture)	Outcome
1. Introduction to Computational Thinking 1.1. Definition, importance, and real-world applications 1.2. Abstraction, decomposition, pattern recognition, and algorithmic thinking	3	1
2. Algorithms and Computability 2.1. Definition, characteristics, and examples 2.2. Time and space complexity, big O notation	3.75	1,2
3. Understanding Logical Reasoning 3.1. Propositional Logic - Syntax, semantics, and truth tables 3.2. Inference Rules and Logical Arguments - Modus ponens, modus tollens, and logical fallacies	3	3,4
4. Problem Decomposition 4.1. Breaking Down Problems - Identifying subproblems and tasks 4.2. Top-Down Design and Stepwise Refinement - Hierarchical problem-solving and iterative development	3.75	5
5. Exploring Problem Analysis 5.1. Identifying Problem Types - Well-defined, ill-defined, and wicked problems 5.2. Problem-Solving Techniques - Brute force, divide and conquer, backtracking, and greedy algorithms	3	6,7
6. Organizing Information: Ordered Structures 6.1. Arrays and Lists - Static and dynamic arrays, insertion and deletion operations 6.2. Stacks and Queues - LIFO and FIFO, implementation, and real-world applications	3	2,7

7. Organizing Information: Unordered Structures 7.1. Sets and Dictionaries - Properties, operations, and use cases 7.2. Hash functions, collision resolution, and performance analysis	3	2,7
8. Computational Creativity 8.1. Concepts, goals, and examples 8.2. Creative Algorithms - Randomness, generative art, and AI-based creative systems	3	2,7
9. Introduction to Recursion 9.1. Recursive Functions - Definition, base case, and recursive case 9.2. Recursion vs. Iteration - Trade-offs, examples, and real-world applications	3	8
10. Introduction to Searching Algorithms 10.1. Linear and Binary Search - Algorithms and performance comparison 10.2. Advanced Search Algorithms - Interpolation search, jump search, and exponential search	3.75	1,2,7
11. Introduction to Sorting Algorithms 11.1. Class 1: Bubble Sort, Selection Sort, and Insertion Sort - Algorithms and performance analysis 11.2. Class 2: Merge Sort, Quick Sort, and Heap Sort - Algorithms and performance analysis	3.75	1,2,7
12. Ethical Considerations in Computing 12.1. Ethical Theories and Frameworks - Utilitarianism, deontology, and virtue ethics 12.2. Privacy, security, AI, AI Bias, and intellectual property	1.5	9

Performance Measures for Evaluation

All assignments are assigned through the Canvas course site. The deadlines are strictly enforced. For example, if the deadline is 11:59 PM, any assignment submitted after this time is considered late. It is also each student's responsibility to submit correct files and ensure the submission is successful before the deadline. If students are unable to submit their assignment through Canvas, they will need to send a copy of their assignment to the instructor before the stated deadline. There will be three exams and each exam will be cumulative with an emphasis on the most recently covered material. Exam details will be posted on the Canvas course site (https://canvas.fiu.edu).

Assignment	Total Points	Percentage of Final Grade
Quizzes (11-Drop-1)	100 each	10%

Knight Foundation School of Computing and Information Sciences
COP 3XXX Computational Thinking

Homework Assignments (3)	100 each	30%
Exam 1	100	20%
Exam 2	100	20%
Exam 3	100	20%
	TOTAL	100%

Letter Grade Distribution Table

Letter	Range%	Letter	Range%	Letter	Range%
A	93 or above	B	82 - 85.9	C	70 - 73.9
A-	90 - 92.9	B-	78 - 81.9	D	60 - 69.9
B+	86 - 89.9	C+	74 - 77.9	F	less than 60

Description of Possible Homework Activities

Homework 1: Logical Reasoning and Algorithms
Description: Students will be provided with several real-world scenarios. For each scenario, they should:
a. Formulate the problem in propositional logic.
b. Design an algorithm (pseudocode accepted) to address the scenario.

Rubric:

Logical Formulation (50 points)	Accuracy: Logical statements correctly reflect the scenario	30 points
	Completeness: All major components of the scenario are addressed	20 points
Algorithm Design (50 points)	Correctness: Algorithm addresses the problem and would produce the desired outcome	25 points
	Clarity: Pseudocode is clearly written and easy to follow	25 points

Homework 2: Problem Decomposition and Solution Design
Description: Students are presented with a complex real-world problem, such as organizing a school event, planning a road trip, or managing a small library. They are required to:
a. Decompose the problem into smaller, more manageable subproblems.
b. Design a step-by-step solution or algorithm (in pseudocode) for each of these subproblems.

For example, if the problem is "Organizing a School Event", subproblems might include "Allocating Budget", "Scheduling", "Resource Management", etc.

Rubric:

Problem Decomposition (50 points)	Clarity: Each subproblem is clearly defined without ambiguity	25 points
	Completeness: All major aspects of the main problem are broken down into subproblems	25 points
Solution Design (50 points)	Relevance: Designed solutions align well with the stated subproblems	25 points
	Detail: Pseudocode or step-by-step processes are comprehensive, considering possible challenges and solutions	25 points

Homework 3: Object-Oriented Programming
Description: Design a simple project that showcases computational creativity. This could be a generative art piece, a randomized story generator, or any creative project leveraging computational techniques.

Rubric:

Concept and Design (50 points)	Originality: The project showcases a unique and novel idea	25 points
	Relevance: The project effectively utilizes computational creativity techniques	25 points
Functionality (40 points)	Code Quality: Code is clean, well-organized, and properly commented	20 points
	Functionality: The project works as intended without errors.	20 points
Documentation (10 points)	Explanation: Clear documentation or write-up explaining the concept, design decisions, and how to run/view the project	10 points