Knight Foundation School of Computing and Information Sciences

Course Title: Computational Thinking **Date:** 6/2/2024

Course Number: COP 3410

Number of Credits: 3

Subject Area: Programming
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 2047 - Python Programming I or COP 2210 - Programming I or COP 2250 - Programming in Java or Advisor's Permission
Corequisite Courses:

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

Prerequisites 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. **Examine** 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** 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. **Explain** the differences and trade-offs between recursion and iteration.
9. **Implement** recursive functions appropriately.
10. **Critically assess** computing solutions, considering ethical implications including such considerations as privacy, security, AI biases, and intellectual property rights.

Association between Student Outcomes and Course Outcomes

BS in Computing: Student Outcomes Graduates of the program will have an ability to:	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 3410 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. <u>Computational Creativity</u> 8.1. Concepts, goals, and examples 8.2. Creative Algorithms - Randomness, generative art, and AI-based creative systems	3	2,7
9. <u>Introduction to Recursion</u> 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. <u>Introduction to Searching Algorithms</u> 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. <u>Introduction to Sorting Algorithms</u> 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. <u>Ethical Considerations in Computing</u> 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	Points Each	Total Points	Percentage of Final Grade
Quizzes (11-Drop-1)	10	100	10%
Homework Assignments (3)	100	300	20%
Exam 1	200	200	20%
Exam 2	200	200	20%
Class Project	300	300	30%
		TOTAL	100%

Knight Foundation School of Computing and Information Sciences
COP 3410 Computational Thinking

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.

Description of Possible Rubric:

Criteria	**Excellent (100)**	**Good (80)**	**Average (60)**	**Below Average (40)**	**Poor (20)**	**Weight**
Logical Formulation						
- Accuracy	Logical statements perfectly reflect the scenario	Logical statements mostly reflect the scenario	Logical statements somewhat reflect the scenario	Logical statements barely reflect the scenario	Logical statements do not reflect the scenario	25%
- Completeness	All major components of the scenario are perfectly addressed	Most major components of the scenario are well addressed	Some major components of the scenario are addressed	Few major components of the scenario are addressed	Major components of the scenario are not addressed	25%
Algorithm Design						
- Correctness	Algorithm perfectly addresses the problem and would produce the desired outcome	Algorithm mostly addresses the problem and would likely produce a good outcome	Algorithm somewhat addresses the problem and might produce a satisfactory outcome	Algorithm barely addresses the problem and is unlikely to produce a satisfactory outcome	Algorithm does not address the problem correctly or would not produce the desired outcome	25%
- Clarity	Pseudocode is perfectly clear and extremely easy to follow	Pseudocode is mostly clear and easy to follow	Pseudocode is somewhat clear and can be followed with effort	Pseudocode is not very clear and is hard to follow	Pseudocode is not clear at all and cannot be followed	25%

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.

Description of Possible Rubric:

Criteria	Excellent (100)	Good (80)	Average (60)	Below Average (40)	Poor (20)	Weight
Problem Decomposition						
- Clarity	Each subproblem is defined with utmost clarity and precision, leaving no room for ambiguity	Each subproblem is clearly defined with minor ambiguities	Subproblems are somewhat clearly defined but with noticeable ambiguities	Subproblems are defined but with substantial ambiguities	Subproblems are not clearly defined, with pervasive ambiguities	25%
- Completeness	All major aspects of the main problem are excellently broken down into detailed subproblems	Most major aspects of the main problem are well broken down into subproblems	Some major aspects of the main problem are broken down into subproblems	Few major aspects of the main problem are broken down into subproblems	Major aspects of the main problem are not adequately broken down into subproblems	25%
Solution Design						
- Relevance	Solutions are perfectly aligned with the stated subproblems, demonstrating deep understanding	Solutions are mostly aligned with the stated subproblems, demonstrating good understanding	Solutions are somewhat aligned with the stated subproblems, demonstrating average understanding	Solutions are barely aligned with the stated subproblems, demonstrating limited understanding	Solutions are not aligned with the stated subproblems, demonstrating lack of understanding	25%
- Detail	Pseudocode or processes are exceptionally detailed, considering a wide range of challenges and solutions	Pseudocode or processes are detailed, considering most potential challenges and solutions	Pseudocode or processes show some detail, considering some challenges and solutions	Pseudocode or processes lack detail, considering few challenges and solutions	Pseudocode or processes are not detailed, not considering challenges and solutions adequately	25%

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.

Description of Possible Rubric:

Criteria	Excellent (100)	Good (80)	Average (60)	Below Average (40)	Poor (20)	Weight
Concept and Design						

Knight Foundation School of Computing and Information Sciences
COP 3410 Computational Thinking

- Originality	The project showcases a highly unique and novel idea, demonstrating exceptional creativity	The project showcases a unique and somewhat novel idea, demonstrating good creativity	The project showcases a moderately unique idea, demonstrating average creativity	The project showcases a somewhat unique idea, demonstrating below-average creativity	The project does not showcase a unique or novel idea, demonstrating poor creativity	20%
- Relevance	The project excellently utilizes computational creativity techniques to a high degree	The project effectively utilizes computational creativity techniques to a good degree	The project utilizes computational creativity techniques to a moderate degree	The project utilizes computational creativity techniques to a limited degree	The project does not effectively utilize computational creativity techniques	20%
Functionality						
- Code Quality	Code is impeccably clean, well-organized, and thoroughly commented	Code is clean, well-organized, and mostly well-commented	Code is somewhat clean, organized, and somewhat commented	Code is somewhat messy, less organized, and poorly commented	Code is messy, disorganized, and not commented	20%
- Functionality	The project works perfectly as intended without any errors	The project works well as intended with minor errors	The project works as intended but with some noticeable errors	The project somewhat works as intended but with many errors	The project does not work as intended and has numerous errors	20%
Documentation						
- Explanation	Documentation is exceptionally clear, providing detailed explanations of the concept, design decisions, and how to run/view the project	Documentation is clear, providing good explanations of the concept, design decisions, and how to run/view the project	Documentation provides a basic explanation of the concept, design decisions, and how to run/view the project	Documentation provides a limited explanation of the concept, design decisions, and how to run/view the project	Documentation is unclear or missing, not adequately explaining the concept, design decisions, and how to run/view the project	20%