LEARNING COMMONS

FRANKLIN UNIVERSITY PROFICIENCY EXAM (FUPE)
STUDY GUIDE

Course Title: ITEC 136: Principles of Programming

Recommended Textbook: https://www.franklin.edu/current-students/academic-resources/textbooks

Number & Type of Questions: Short answer, essay problems, and programming questions

Permitted Materials: ITEC 136 – Final exam cheat sheet is included in exam

Time Limit: 120 minutes (2 hours)

Minimum Passing Score: 75%

Exam Format: The exam consists of short answers, essays, problems, and programming questions. Passing is 80%. It is two hours in length and is closed book, and notes. For those students where English is a second language, a translation dictionary may be used. Otherwise, no external materials allowed.

Language: The exam currently uses Python as its programming language. A review of these languages would be useful

Notes: You will be expected to do problems related to both reading and writing code. Scoping rules, arrays, variables, control structures, and general computer concepts such as bits and bytes, number systems, etc. are all fair game. The outcomes below contain more detail about the exam contents.

Format varies

Outline of the Topics Covered:

Course Description

This course introduces programming to individuals with little or no programming background. The goal of this course is to introduce the fundamentals of structured programming, problem solving, algorithm design, and software lifecycle. Topics will include testing, data types, operations, repetition and selection control structures, functions and procedures, arrays, and topdown stepwise refinement. Students will design, code, test, debug, and document programs in a relevant programming language.

Course Outcomes

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

1. Explain the stages of the software lifecycle.
2. Design solutions using top-down stepwise refinement.
3. Translate algorithms into clearly documented and modularized programming language code.
4. Generate and execute industry accepted test plans.
5. Use sound object-oriented design principles to implement linear data structures.
6. Apply recursion to solve applicable problems.
7. Analyze the efficiency of algorithms.
8. Describe and implement well-known searching and sorting algorithm

Course Content

Week 1

1. Apply general problem-solving strategies to develop computer algorithms
2. Produce working programs which includes operators, variables, branching, functions, loops
3. Correct programs by using debugging

Week 2

1. Articulate the principles of effective software design
2. Use stepwise refinement to solve problems
3. Employ test driven methodologies to ensure correctness

Week 3

1. Evaluate string comparison operators according to lexigraphical order
2. Demonstrate slicing operations on strings
3. Construct simple algorithms for string pattern matching

Week 4

1. Articulate the difference between procedural programming and object-oriented programming
2. Model an object's state and methods
3. Implement a class
4. Describe the purpose and use of an initializer method
5. Distinguish a member function from a nonmember function

Week 5

1. Use a class to represent an abstract mathematical concept e.g. fractions complex numbers etc.
2. Use mutator method
3. Distinguish between deep equality and shallow equality

Week 6

1. Define properties of lists
2. Employ common list manipulation techniques e.g. access, insert, remove, concatenate, slice
3. Process lists using loops
4. Replace the loops with list comprehensions
5. Distinguish lists from tuples

Week 7

1. Compare and contrast iteration and recursion
2. Identify and explain three basic components of recursive algorithms
3. Employ recursion to solve simple relevant problems

Week 8

1. Use recursion to locate files
2. Read, write, and update files
3. Process records in a file

Week 9

1. Use unordered associative collections
2. Explain operations on dictionaries
3. Use dictionaries to solve a relevant problem

Week 10

1. Analyze the efficiency of algorithms
2. Explain asymptotic notation
3. Distinguish between big O, big Theta, and big Omega notation
4. Iteratively improve the efficiency of an algorithm

Week 11

1. List the typical operations and properties of linear data structures e.g.
2. stacks, queues, dequeues and linked lists
3. Determine the appropriate data structure for a problem
4. Use linear data structures to solve relevant problems

Week 12

1. Compare, contrast, and demonstrate the execution of the algorithms for selection, insertion, and bubble sorts
2. Analyze the performance of selection, insertion, and bubble sorts
3. Compare, contrast, and demonstrate the execution of the algorithms for linear and binary search
4. Analyze the performance of linear and binary search