Course Assessment Document for CS 151 Computational Thinking

Departmental Outcomes

1. Proficiency in computational thinking
2. Ability to analyze systems at the three levels of computer science: theory, software, and hardware
3. Proficiency in the design and implementation of algorithms using multiple programming languages
4. Ability to apply computational thinking to a diverse set of problems and disciplines
5. Ability to communicate effectively and collaborate with others
6. Ability to adapt to new challenges and computational environments

Course Description

The course is an introduction to computational thinking: how we can describe and solve problems using a computer. Using the Python language, students will learn how to write algorithms, manipulate information, and design programs to make computers useful tools. Through lectures, weekly homework and quizzes, and weekly programming projects, students will learn about abstraction, how to divide and organize a process into appropriate components, how to describe processes in a computer language, and how to analyze and understand the behavior of their programs. Students will communicate the results of their work through project reports.

Prerequisites: None

Rationale for prerequisites: The course is intended for students with little or no experience with computer science or programming. Enrollment in the lab sections–two per lecture section–is capped at 22 to match the number of available computers. Priority is given to majors and minors and then reverse seniority because this course is a prerequisite for all other computer science courses.

Rationale for Distribution: The course satisfies the college Q distribution requirement. Abstractions of objects, concepts, and ideas are core to computer science, especially in the design of computer programs. Students in the course are required to create such abstractions and implement them as computer programs. The completion of such programs requires analytical and quantitative reasoning.

Desired Course Outcomes

A. Students can read a simple program and correctly identify its behavior

B. Students can convert a problem statement into a working program that solves the problem.

C. Students understand abstraction and can break down a program into appropriate procedural and object-oriented components

D. Students can generate an approximate model of computer memory and describe how an algorithm affects its contents.

E. Students can communicate the result of their work and describe an algorithm.

We will disseminate the desired course outcomes to students via the course web page, syllabus and in class.

Course Matrix

Outcome	Activities	Method of Assessment	Departmental Outcome
A	Lectures, Homework	Exams, quizzes	1, 3
B	Homework, Labs, Projects	Exams, quizzes, graded labs, graded projects	1, 2, 3, 4, 6
C	Lectures, Labs, Projects Homework	Exams, quizzes, graded labs, graded projects	1, 3, 4, 6
D	Lectures, Homework	Exams, quizzes	2, 3, 6
E	Project reports	Graded projects	5

Grade Calibration Matrix

Outcome	Meaning of the grade A
A	The student can understand programs given to them and figure out their behavior including boundary cases.
B	The student can create a working program that solves the problem using the exact specification given and which may exceed the minimum required capabilities.
C	The student can create functions with appropriate parameters and classes with appropriate methods to model the problem in a reusable way, possibly generalizing the code to solve a more general problem.
D	The student can understand how data resides in memory and the implications of the memory model in terms of efficiency as well as correctness.
E	Reports are well-written, concise, and clear. The reports clearly describe the project and effectively use example code to demonstrate concepts and design decisions. The reports are written so that students outside the course could understand them.

Outcome	Meaning of the grade B
A	The student can understand programs given to them and figure out their behavior in most cases.
B	The student can create a working program that solves the problem but not following the exact specification given.
C	The student can create functions with appropriate parameters and classes with appropriate methods to model the problem given.
D	The student can understand how data resides in memory and the implications of the memory model in terms of correctness.
E	Reports are well written and clear. The reports describe the main points of the project, mention design issues and may make use of example code to support the text. The reports are written so that students taking the course could understand the work.

Outcome	Meaning of the grade C
A	The student can understand most programs given to them and figure out their behavior in most cases.
B	The student can create a working program that solves the general case of a problem but not all special cases and not following the exact specification given.
C	The student can create functions and classes with appropriate methods to approximately model the problem given.
D	The student can understand how most data resides in memory.
E	Reports describe the work, but may take the form of a narrative of the student's difficulties rather than focusing on the results. The reports assume the reader has in-depth knowledge of the project.

Outcome	Meaning of the grade D
A	The student can understand only some of the programs given to them and only in the general case.
B	The student can create a working program but it solves a different problem than the given one.
C	The student can create functions and classes but the organization of them is inappropriate to model the problem given.
D	The student can understand how some data resides in memory.
E	The student's reports are incomplete or not well written and may contain errors. The report includes some description of the student's work, but little explanation.

A student who receives an F does not meet the criteria for a D or any higher grade.