E116: Intro to Programming and Algorithmic Thinking

School: SES/ECE

Owning Academic Unit: ECE

Course Title: Intro to Programming & Algorithmic Thinking

Proposed Course # or Level: Freshman

Catalog Description:

The objective of this course is to introduce students to computer programming and algorithmic thinking. This course is taken by incoming freshmen who typically have limited exposure to computer programming. The course works with it's participants to provide: (i) a solid foundation in computer programming, (ii) an understanding of algorithmic thinking, and (iii) numerous tools, techniques and foundations necessary for the student to continue to build these skills as they journey through a variety of engineering disciplines.

Course Objectives:

Beyond helping students learn programming, the course prepares them to apply computing and algorithmic thinking in a variety of engineering disciplines. Recognizing the increasingly important role computing plays in domains such as mechanical, civil, bio-tech, software, electrical and computer engineering, this course aims to provide both an introduction as well as a foundation necessary to rapidly build skill sets to bring computing to all aspects of engineering. The course also has a strong project component, which lets the student apply skills to a real problem, as well as work in teams to accomplish complex tasks.

List of Course Outcomes:

After taking E116, a student should be able to demonstrate the following:

1. Students understand and are comfortable with integrated development environments (IDEs) typically required to build computer programs. This includes editors, compilers, and tools to build and run software.
2. Students understand the structure of a program, and are able to take a problem-solution formation description and convert this to a computer program.
3. Student is able to understand and apply principles of algorithmic thinking, which includes understand the problem, mapping out a solution space, picking an optimal solution, and measuring and reporting the goodness of a given solution.
4. Students understand that data drives programs forward, and are comfortable with creating different kinds of variables, storing values in variables, and using variables in mathematical equations typical in engineering.

5. Students are able to use conditional logic to implement decision making in programs, and dynamically alter the execution of code depending on conditions.
6. Students are able to apply iteration to process data -- this includes for-loops and while-loops.
7. Students are able to think of algorithms and solutions in a modular way, and are comfortable defining and using functions in a programming laguage.
8. Students are able to design complex systems using object oriented programming (OOP) principles, by defining and using classes and objects.
9. Students are able to read and write data to files that are external to the program and resident on a hard-drive
10. Students are able to work in teams and accomplish a real world project

Prerequisites: N/A

Co-Requisites: N/A

Cross-listing: N/A

Number of credits: Four (4)

For Undergraduate Credit toward Degree:
☐ **Yes** ☐ No ☐ Not for Dept. Majors ☐ Other

Is this course repeatable for additional credit? ☐ Yes ☐ **No**

Mode(s) of Delivery: ☐**In-person** ☐Online ☐Hybrid ☐Other ______________

For instructional format, you may select more than one. If a course includes multiple formats (for example: lecture and lab; lecture, lab and recitation, etc.), please indicate which controls grading.

Lecture/Lab Combo is when both the lecture and lab are scheduled in the same block. For Lecture/Lab Combo, you do not need to indicate which controls grading.

Instructional Format: ☐Lecture ☐Lab ☐Recitation ☐**Lecture/Lab Combo**

If multiple instructional formats are selected, please indicate which controls grading:

Labs control grading -- the TAs actively grade assignments, midterms and final projects. Lectures are primarily used to disseminate new learning every week. At the end of the lecture, an assignment is released to the students, which is due the following week after the labs. During the labs/recitations, the TAs expand on the topics introduced in lecture, actively help students with the assignment, and subsequently grade the submissions. Student work is graded for every lab section on a weekly basis. At the end of the term, grades from every lab section is combined to assign a final letter grade to every student.

Typical Period(s) Offered: ☐**Fall** ☐Spring ☐Summer A ☐Summer B

Effective date: **Fall 2021**

Contact Person(s) (In the case of joint ownership, please list a contact for each department/school.)
Name: Mukundan Iyengar
Title: Teaching Associate Professor, ECE
Phone number:
Email address: miyengar@stevens.edu

Textbook(s) or References (List required and recommended texts including publisher and year in a recognized format such as APA, AIP, Chicago or MLA):

C++ Primer -- S. Lippman, J. Lajoie, and B. Moo, *Fifth Edition,* ISBN-13: 978-0321714114

Grading Percentages: **HW** □ **Class work** □ **Mid-term** □ **Final** □ **Projects** □

Other □ (specify both percent and kind of work)

Homeworks: 60%

Midterm (I & II): 30%

Final Project: 10%

Sample Syllabus: This syllabus should be sufficiently detailed to allow the Curriculum Committee to understand and discuss the scope of the course, its aims and assignments. The homework and reading sections should provide sufficient detail for the Committee to judge the amount and kind of work required of students. The Committee understands that this syllabus is a sample of how a course might be organized, not a commitment to always offer the course exactly as described every time. Note that a syllabus is not merely a listing of topics or a restatement of the catalog description.

	Topic(s)	**Reading(s)**		**HW**
Week 1	Intro to course, grading, TAs, logistics.	Chapter 1.1, 1.2 and 1.3		Install compiler. Write “hello world”
Week 2	Basics of coding (what is compiler?) Basic structure of a program (hello world). **Variables**: Storing and manipulating data.	Chapter 2.1, 2.2, 2.3 and Chapter 3.2		Integer, Float, Double, Char, string variable types. Accept input and print output
Week 3	**Conditional Statements** (if/else)	Chapter 1.4.4		Write a program that determines final “grade” based on input data
Week 4	**Loops** and iterations	Chapter 1.4.1 and 1.4.2		Write a program to calculate mean and standard deviation of a large input
Week 5	**Arrays**: Storing and manipulating large data	Chapter 3.5 and 3.6		Write a program to capture a large input and store and process it using arrays.

Week 6	**Functions:** Composable code components. Defining, using and returning values	Chapter 6.1, 6.2 and 6.3		Write a realistic calculator that uses functions to perform a variety of tasks.
Week 7	**Midterm-I**			
Week 8	**Classes/Objects**: Advanced computer programming	Chapter 7.1, 7.2, 7.3 and 7.4		Write a program to mimic the actions of a vending machine using classes and objects
Week 9	**Final project:** Requirements, coding, testing and documentation. Overview and scope.	Online resources, tutorials and sample code		Design and deliver final project.
Week 10	**File Operations:** open, read and write into/from files using C++	Chapter 8.1 and 8.2		Write a program to parse through text in a file and create a phone book application
Wee 11	**Intro to algorithmic thinking part-I:** Measuring the goodness of code. Efficiency and computational time.	Instructor provided lecture notes		Benchmark the performance of a given solution in terms of execution time and memory consumption
Week 12	**Intro to algorithmic thinking part-II:** Good v/s bad algorithms, *A Case Study* (Matrix related)	Instructor provided lecture notes		Write a program to perform arithmetic of large matrices, and benchmark performance
Week 13	**Midterm-II (**open office hours for help with project)	N/A		
Week 14	Final Project submissions and wrap up.	N/A		Finals Projects submitted.