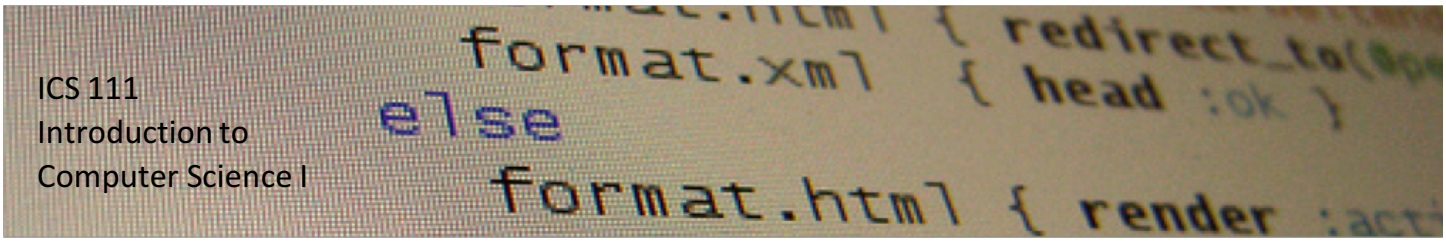

ICS 111 - Introduction to Computer Science I

Instructor Information

David Maxson
David.Maxson@hawaii.edu
Office hours: Online

Windward Community College Mission Statement

Windward Community College offers innovative programs in the arts and sciences and opportunities to gain knowledge and understanding of Hawai‘i and its unique heritage. With a special commitment to support the access and educational needs of Native Hawaiians, we provide O‘ahu’s Ko‘olau region and beyond with liberal arts, career and lifelong learning in a supportive and challenging environment — inspiring students to excellence.

Catalog Description

Intended for Computer Science majors and all others interested in a first course in programming. An overview of the fundamentals of computer science emphasizing problem solving, algorithm development, implementation, and debugging/testing using an object-oriented programming language.

Student Learning Outcomes

The Student Learning Outcomes for this course are:

- Use an appropriate programming environment to design, code, compile, run and debug computer programs.
- Demonstrate basic problem solving skills: analyzing problems, modeling a problem as a system of objects, creating algorithms, and implementing models and algorithms in an object-oriented computer language (classes, objects, methods with parameters, abstract classes, interfaces, inheritance and polymorphism).
- Illustrate basic programming concepts such as program flow and syntax of a high-level general purpose language.
- Demonstrate working with primitive data types, strings and arrays.

Student Learning Outcomes Alignment

Student Learning Outcome	Lessons and Assessments
Use an appropriate programming environment to design, code, compile, run, and debug computer programs.	Lessons 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 Assignments 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
Demonstrate basic problem solving skills: analyzing problems, modeling a problem as a system of objects, creating algorithms, and implementing models and algorithms in an object-oriented computer language	Lessons 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 Assignments 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
Illustrate basic programming concepts such as program flow and syntax of a high-level general purpose language.	Lessons 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 Assignments 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
Demonstrate working with primitive data types, strings, and arrays.	Lessons 3, 4, 5, 6, 7, 8 , 9, 10, 11, 12, 13, 14 Assignments 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

Course Content

Concepts	Skills
1. Use an appropriate programming environment to design, code, compile, run and debug computer programs. a. Programming-tools. 1) Integrated Development Environment (IDE) or a text editor and command line-based compilation and execution. b. Coding a solution. 1) Self-documenting programs. 2) Good formatting. c. Compile and run programs. a. Debug programs.	1. Use an appropriate programming environment to design, code, compile, run and debug computer programs. a. Use programming tools to model a problem and design algorithms that express its solution. b. Formulate models and algorithms in the syntax of an object-oriented programming language using either an Integrated Development Environment (IDE) or a text editor. c. Utilize either an IDE or a command prompt to compile and run programs. d. Test and debug programs to produce code that runs and generates the correct results.
2. Demonstrate basic problem solving skills: analyzing problems, modeling a problem as a system of objects, creating algorithms, and implementing models and algorithms in an object-	2. Demonstrate basic problem solving skills: analyzing problems, modeling a problem as a system of objects, creating algorithms, and implementing models and algorithms in an object-oriented

oriented computer language (classes, objects, methods with parameters, abstract classes, interfaces, inheritance and polymorphism). a. Analysis of a problem by identifying objects and classifying them. b. Design a solution to the problem by defining the messages objects send each other, the parameters the messages carry and the inheritance among object classes. c. Classes, objects, and methods. 1) Classes objects, and methods described. a) Classes. b) Objects. c) Method declarations and method calls d) Overloaded methods. 2) Incorporate parameter passing. a) Formal and actual parameters. b) Returning values from methods c) Parameter passing by value and by reference. 3) Write simple classes and objects. a) Classes. b) Objects. c) Method declaration/implementation and method calls. d) Constructors. e) Encapsulation through visibility modifiers (public, private) f) Class and instance methods and fields (static) 4) Inheritance and Polymorphism a) Extending classes, subclasses b) Overriding methods c) Polymorphism 5) Interfaces a) Interfaces as types b) Implementing by classes 6) Program Development a) Algorithm design and representation using pseudocode, flowcharts, etc. b) Evaluate algorithm efficiency. c) Stepwise refinement. d) Program lifecycle.	computer language (classes, objects, methods with parameters, abstract classes, interfaces, inheritance and polymorphism). a. Classes, objects, and methods 1) Use API classes, objects, and methods, citing examples. 2) Write simple classes and create objects that interact between multiple classes. 3) Understand parameter passing and methods returning values 4) Inheritance and Polymorphism a) Model a problem as a hierarchy of classes b) Differentiate between overloading and overriding. 5) Define Interfaces and implement them with classes b. Apply problem-solving techniques such as stepwise refinement and object-oriented analysis c. Incorporate the concept of software life cycle into program development. d. Determine and design an algorithm to solve a specific problem. e. Evaluate algorithm performance.
3. Illustrate basic programming concepts such as program flow and syntax of a high-level general purpose language. a. Sequence.	3. Illustrate basic programming concepts such as program flow and syntax of a high-level general purpose language. a. Describe sequential, branching, and repetitive

b. Selection. c. Repetition.	concepts. b. Use flowcharting to capture sequential, branching, and repetitive concepts. Incorporate good programming practices
4. Identify relationships between computer systems, programming and programming languages. a. Computer organization and architecture (memory, arithmetic-logic unit, control unit). b. Binary representation of data (range of data type, precision and round-off, image representation). c. Operating system concepts. d. Programming language assembler/compiler.	4. Identify relationships between computer systems, programming and programming languages. a. Examine the hardware (binary numbers, character encoding, Boolean logic) and basic computer system architecture concepts. b. Examine system software and virtual machine concepts. c. Describe the concept of program compilation and translation to machine code.
5. Demonstrate working with primitive data types, strings and arrays. a. Primitives Types 1. Numeric, character and boolean types. 2. Numeric accuracy. 3. Memory requirements. 4. Declaration. 5. Initialization. b. Integer Arithmetic 1. Addition and subtraction, increment and decrement 2. Multiplication, division, and modulo. 3. Truncation. c. Casting 1. Type assignment. 2. Implicit and explicit casting. d. Strings 1. Constants 2. Concatenation. e. Arrays 1. Declaration 2. Access to array vs. access to an element 3. Multidimensional arrays	5. Demonstrate working with primitive data types, strings and arrays. a. Primitive types 1) Utilize and understand primitive types, their accuracy, memory requirements 2) Declarations and initialization of primitive types. 3) Demonstrate integral arithmetic including mod. 4) Explain casting and differentiate between implicit and explicit casting. b. Strings c. Arrays

Course Tasks

In this class, you must show mastery of each concept by completing a projects. Each assignment is worth 6 points except for the final project, which is worth 18 points.

Points are awarded as follows:

- 6 points - All aspects of the assignment are met. This not only includes the core concept from the lesson, but clear, well written code. Clear, well written code means code that is readable and includes comments and whitespace with no unused variables or instructions.
- 5 points - All aspects of the assignment are met and there is no unnecessary code, but prompts and messages are not user friendly or the code is not easy to read due to a lack of comments and/or whitespace..
- 3 - 4 points - The program compiles and the core concept from the lesson is used correctly, but the code is hard to read and/or includes unnecessary code. Some of the instructions for the assignment weren't followed.
- 1 - 2 points - The program compiles but the core concept from the lesson is only partially used or used incorrectly. Some of the instructions for the assignment weren't followed.
- 0 points - The program doesn't compile, there are runtime errors, or the core concept from the lesson wasn't used. Some or all of the instructions for the assignment weren't followed.

If there are error(s), then I will let you know and return the assignment to you. You should then correct the assignment and resubmit it.

Every assignment has a due date. To get full credit for the assignment it must be submitted by the due date. You may turn them in late but there will be a one point penalty if you do. You may resubmit projects after they are returned to raise your grade (although you will not be able to recoup the point lost for turning it in late). There is no late penalty for resubmissions, only original submissions. The final deadline for all submissions and re-submissions is Dec. 8. This is a hard deadline and no assignments will be accepted after that date.

Assignment Tasks and Grading

Your final grade will be determined by the number of assignments you complete. Each assignment is worth 6 points except for the Final Project, which is worth 18 points. There are 12 assignments and the Final Project for a total of 90 points. The grade scale for your letter grade is:

- A: 81 - 90 points
- B: 72 - 80 points
- C: 63 - 71 points
- D: 45 - 62 points
- F: 0 – 44 points

Learning Resources

We will be using an online resource called Revel in this course. The Revel site has an electronic copy of the textbook as well as videos and "check your understanding" quizzes. It will be the main source of information about the Java programming language. It is essential for you to have access to the site. Note that this is only a tool to help you learn the Java programming language. All due dates and other course requirements are listed in the Laulima course site.

We will use Laulima for submitting and returning all assignments. All grades will be posted in Laulima and you will be able to track your progress by utilizing the gradebook. You will be able to post and read questions and comments on the discussion boards. Use the private message tool in Laulima to contact the instructor.

We will be using the Java programming language to develop our programs. Go to the Java Download Page (http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) to download the latest SE JDK. It is vital that you install it correctly, including setting the correct path environment
Although it isn't necessary, I also recommend you use an Integrated Development Environment such as jGrasp (http://jgrasp.org). I do not recommend using NetBeans or Eclipse at this time. Both insert code in your projects that could keep it from compiling from the command line.

As an alternative, you can use the uhunix (type ssh username@uhunix.hawaii.edu at the command line. Replace username with your UH username) or an online development site such as Cloud9 IDE (https://c9.io).

Other resources

Tutoring may be available from the TRIO office in the Library Learning Commons on the WCC campus.

Policies

Disabilities Accommodation Statement
If you have a physical, sensory, health, cognitive, or mental health disability that could limit your ability to fully participate in this class, you are encouraged to contact the Disability Specialist Counselor to discuss reasonable accommodations that will help you succeed. Ann Lemke can be reached by phone at 235-7448, by email at lemke@hawaii.edu, or by stopping by her office in Hale 'Akoakoa 213 for more information.

Academic Dishonesty - Cheating and Plagiarism
You are responsible for the content and integrity of all work you submit. The guiding principle of academic integrity will be that all files, work, reports, and projects that you submit are your own work.

You will be guilty of cheating if you:

- Represent the work of others as your own (plagiarism).
- User or obtain unauthorized assistance in any academic work.
- Give unauthorized assistance to other students.

- Modify, without instructor approval, an examination, paper, record, or report for the purpose of obtaining additional credit.
- Misrepresent the content of submitted work.

Netiquette
Whenever you post something to the discussion board or other online forums, you are expected to follow proper netiquette. Be respectful at all times. Do not use obscene language or make disparaging comments, even if it is meant as a joke. Remember that others cannot see your facial expression nor hear your tone of voice, so they will not know you are trying to be witty. Do not use all caps. Using all caps is normally interpreted to be shouting.

Discussion Boards
Discussion boards are to be used for class work only. Do not post political or other comments or statements, nor solicit sales for any type of product. Your instructor will be monitoring all communication in Laulima and will take appropriate action when necessary

A Final Thought

All programming languages use the same basic concepts. By learning the concepts and writing your initial program in pseudocode, you should be able to use any programming language to code your software. It is vital that you understand these concepts. You will use them throughout your studies in Computer Science and as a programmer or Software Engineer afterward. The best way to learn them is to use them. Try designing and creating programs that you will find useful. Good luck!