**Course Title:** How to Think Like a Programmer

Proposed by: Larry Lambert
Programming and Web Development
Minuteman High School

**Course Title**

How to Think Like a Programmer

**Description of Course**

This course is designed to provide students with the fundamental background, knowledge and skills necessary to optimize their ability to be successful as they embark on a course of study in computer programming. As they successfully complete the course, students will enhance their capabilities in math and science, build strategic-thinking skills, develop strong oral and written communication skills and develop their ability to work in a team environment.

**Course Rationale**

As the computer has become ubiquitous in our daily lives, more and more students have begun to experience the desire not only to consume but also to create digital media. As these students embark on the road to becoming computer programmers, they fail to recognize the underlying proficiencies required to successfully attain competency at computer programming. Over the past ten years, I have noticed that many of the students who enter the program lack the requisite mathematics background to fully comprehend what is necessary to come up with an algorithm that will successfully solve the problem that they are being asked to create a program to solve.

This problem is not unique to students participating in the Programming and Web Development program at a Career Technical Education high school. In discussing the problem with other computer science teachers at a workshop this past summer, I learned that many Computer Science professors, at 4-year colleges, encounter similar issues with their incoming freshman. The establishment of this course at the high school level will allow our students to achieve a higher competency level prior to their graduation from high school and provide them with a foundation that will support their long term aspirations at post-secondary institutions.

In assessing the programming skills of current upperclassman and last year's seniors, it seems that previous instructors fell into the trap of believing that students with the ability to write programs in a larger number of languages are good programmers. It is as though they believe that a student who programs poorly in Visual Basic will somehow develop better programming techniques once they begin to learn Python, Java or C++. The truth of the matter is that a student who has not grasped the basic concepts of computer programming while learning their first language will only continue with poor programming practices as they go on to a second, third and subsequent language.

In order to prepare students to embark on successful academic career in computer programming, they must have a fairly robust understanding of mathematics. Therefore, the first step is to assess the students' capabilities and when necessary provide each student with the required level of remedial support for them to understand how to "solve the puzzle" by constructing the proper equations to yield the correct solutions. In this way, developing the plan is a lot like doing math word problems. It doesn't stop there. The typical mathematics background must then be supplemented with Boolean algebra. Because computers are built as collections of switches (transistors) that are either "on" or "off," Boolean algebra is a very natural way to represent

digital information. Once students have reached the necessary level of proficiency in mathematics, they are ready to go on to algorithms.

Algorithms have been commonly defined in simple terms as "instructions for completing a task". They've also been called "recipes". Algorithms are indeed instructions. Perhaps a more accurate description would be that algorithms are patterns for completing a task in an efficient way. They are the sequence of steps or operations that must be carried out in order to generate a solution to a problem. Students must learn how to formulate some relatively complex algorithms to solve problems that on the surface may appear simplistic. The students need to understand how to interpret the given problem before they can represent the correct solution and effectively use specific tools or techniques

The final step that students must develop as "pre-programmers" is the ability to document their ideas. This can be done in either of two ways; a flow chart or pseudo-code. These documents will help the student clarify what they want their program to accomplish and verify that the inputs, processes, decisions and outputs that are to be implemented match those of the problem to be solved. As a student begins to document their proposed solution, they may uncover some special case or some other hole that they neglected to consider.

All of these skills are essential to the development of a successful programmer and are often overlooked if we take a short-sighted approach to the teaching of programming. If we can avoid the myopic view that it is all about the number of languages and look further down the road, we will be planting a seed with these students that will yield a stronger crop of programmers. The course, Think Like a Programmer, would provide fertile soil for this endeavor.

**Course Objectives**

After completing the course, students will be able to:

1. Demonstrate the rudimentary mathematics skills required to find the solutions to word problems.
2. Select the appropriate theoretical concepts for solving word problems.
3. Correctly set up mathematical equations based on the information provided within the problem.
4. Read and comprehend a problem statement (word problem), then synthesize the problem into a mathematical equation or sequence of equations necessary to solve the problem.
5. Formalize their solution strategy (algorithm) and represent it; first graphically as a flow chart and then in the form of pseudocode.
6. Implement the algorithm represented by the pseudocode in the Python programming language.
7. Execute the code in the Python interpreter.

**Framework Standards**

Massachusetts Vocational Technical Education Framework
Information Technology Services Cluster
Programming and Web Development
August, 2007

| VTE # | Acad # | Standard | ILO Connection |
|---|---|---|---|
| 3.B.02c | 7.M.2 | Given the formulas, convert from one system of measurement to another. Use technology as appropriate. | ILO 1, ILO 2, ILO 3, ILO4 |
| 3.B.03c | 8.M.2 | Given the formulas, convert from one system of measurement to another. Use technology as appropriate. | ILO 1, ILO 2, ILO 3, ILO4 |
| 3.B.04c | 8.N.1 | Compare, order, estimate, and translate among integers, fractions and mixed numbers (i.e., rational numbers), decimals, and percents. | ILO 1, ILO 2, ILO 3, ILO4 |
| 3.B.05c | 7.P.4 | Solve linear equations using tables, graphs, models, and algebraic methods. | ILO 1, ILO 2, ILO 3, ILO4 |
| 3.B.06c | 7.P.6 | Use linear equations to model and analyze problems involving proportional relationships. Use technology as appropriate. | ILO 1, ILO 2, ILO 3, ILO4 |

**2.N Explain fundamental programming theory**.

| VTE # | Standard | ILO Connection |
|---|---|---|
| 2.N.06 | Design program logic using graphical techniques (flow charts). | ILO 5 |
| 2.N.07 | Design program logic using pseudocode techniques. | ILO 5 |
| 2.N.08 | Identify the use of program design tools. | ILO 4, ILO 5, ILO 6 |
| 2.N.09 | Explain structured/modular programming. | ILO 5, ILO 6 |

**2.P Develop programs**.

| VTE # | Standard | ILO Connection |
|---|---|---|
| 2.P.01 | Develop programs using desired language. | ILO 6, ILO 7 |
| 2.P.02 | Develop programs that use arithmetic operations. | ILO 1, ILO 2, ILO 3, ILO 4, ILO 6, ILO 7 |
| 2.P.03 | Develop programs that use relational operators. | ILO 6, ILO 7 |
| 2.P.04 | Explain and apply the use of logical operators. | ILO 6, ILO 7 |
| 2.P.05 | Explain and apply compound conditions. | ILO 6, ILO 7 |
| 2.P.06 | Explain and apply control breaks. | ILO 6, ILO 7 |
| 2.P.07 | Explain and apply methods of calculating subtotals and final totals. | ILO 6, ILO 7 |
| 2.P.08 | Explain and apply iterative and conditional loops. | ILO 6, ILO 7 |

**Course Benefit to Students**

The course, How to Think Like a Programmer, is intended for freshman students in the Programming and Web Development technical area. As a result, the benefits derived from the material presented in the course will prove advantageous to the students throughout their high school careers and beyond. The positive impact of the course to students who successfully complete it will be evident both within the technical area and in the academic classroom.

Since the initial focus of the course is the evaluation and remediation (when necessary) of each student's math skills, there should be a direct benefit observed in the student's performance in both math and science courses. In addition to these obvious connections, students will acquire other skills that will prove beneficial in other aspects of their high school careers. Students will develop the ability to split problems into discrete components that will make them easier to solve. They will also learn to organize their thoughts and develop strategies to tackle complex problems. In discussions with other vocational instructors and academic teachers, I hear a recurrent theme regarding students who lack simple math and science skills. Multiple vocational and academic teachers have spoken of students who are unable to read a ruler; many times adding the thought that it is because they don't understand fractions.

The greatest benefit to be gained by the addition of this course to the Programming and Web Development curriculum will be the performance of the students on programming activities within the technical area. Currently, far too many students struggle with the question of "How do I approach the problem?" The students need to understand how to interpret the given problem before they can represent the correct solution and effectively use specific tools or techniques. When they are asked to complete a sequence of operations on a collection of data, many have a lot of difficulty organizing their thoughts and develop what is often referred to as "spaghetti code." The image that this phraseology invokes is spot on; the code is a bent and twisted mess that makes it impossible to discern one end from another. These students spend long periods of time writing code that never quite works. In observing students, it becomes readily apparent that they become frustrated and amass hundreds of lines of code when 25 or so lines could have gotten the job done if only they had planned.

Once students successfully complete the How to Think Like a Programmer course they will have developed the ability to look at a problem and see a clear path from user input through data processing and on to program output. They will be well practiced at making the most of code reuse with functions, classes, and libraries. Picking the perfect data structure for a particular job will be second nature and they will be ready to master more advanced programming tools like recursion. These are skills that will serve them well in high school and beyond.

Students who graduate from a Programming and Web Development Chapter 74 program will more likely than not continue their education at the post-secondary level. As they do, they will continue to derive benefit from the material covered in the How to Think Like a Programmer course. The training in logical thinking will position students to become strong performers as they progress through their college careers, whether they pursue degrees in Computer Science or some other discipline. Having learned to approach problem solving in a methodological way is a portable skill that will prove useful no matter what field the student decides to pursue.