



GREAT NECK PUBLIC SCHOOLS

"Where Discovery Leads to Greatness"

#DiscoverGreatness

Phipps Administration Building
Office of Instruction
345 Lakeville Road
Great Neck, New York 11020

516.441.4009
516.441.4010
Fax 516.441.4992

Curriculum Profile: Computer Science 2

<u>Department</u>	Mathematics	
<u>Course Name</u>	Computer Science 2	
<u>Course Length</u>	1 Semester	
<u>High School Credits</u>	0.5	
<u>Description</u>	This course serves as the immediate follow up for Computer Science I – transitioning into more challenging math and conceptual topics. Successful completion of Computer Science I is a pre-requisite. Students will have more responsibilities throughout the course on the academic side when compared to Computer Science I. This course will use the Java programming language.	
<u>Target/eligible students</u>	Grades 9-12, must be enrolled in or have completed Algebra 1 and successfully completed Computer Science 1.	
<u>CSTA Learning Standards Link(s)</u>	Standards Computer Science Teachers Association (csteachers.org)	
<u>Primary texts and materials</u>	Provided by teacher	
<u>Scope/Sequence</u>	• Overview /Standards	Students will: explore Computer Science from a conceptual and mathematical perspective while using coding as a tool to further their understanding of key concepts. This includes exploring math in different number bases (binary, decimal, hexadecimal) and learning all basic algebraic operations, positive and negative numbers, floating-point (fractional) numbers. Students will learn how these bases can be used to construct other systems such as ASCII for text representation. Students then transition into learning the inner mechanics of Java/OOP (object-oriented programming). Advanced and complex logic as well as encryption algorithms are explored towards the end of the course. Emphasis on structured and efficient software testing is investigated. A final project consists of mathematical research aided by software simulations.
<u>Unit 1: Number Bases/Math</u>	• 2-DA-07 • 3A-CS-02 • 3A-DA-09	The course begins with students “relearning” what they know about basic math and algebra. Computers/digital electronics operate in binary – while humans operate in decimal. Students explore addition, subtraction, multiplication, negative numbers, and fractional numbers in binary to get insight into how electronic components process information. Digital representations of data such as text, images and audio are also investigated. • Represent data using multiple encoding schemes. • Compare levels of abstraction and interactions between application software, system software, and hardware layers. • Translate between different bit representations of real-world phenomena, such as characters, numbers, and images.
<u>Unit 2: Java & Object-Oriented Programming</u>		The programming language for the course – Java – is formally introduced. Syntax, structure, and object-oriented ideas are investigated and explored in a conceptual way. Students will revisit many problems from Computer Science I and consider them from a new and more experience perspective. Concepts such as factoring

	• 3B-AP-14 • 2-AP-14 • 2-AP-13	code, object attributes, constructors and functions are explored in depth. Monolithic problems are converted into OOP solutions with pros/cons discussed. • Construct solutions to problems using student-created components, such as procedures, modules and/or objects. • Create procedures with parameters to organize code and make it easier to reuse. • Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
<u>Unit 3: Advanced Logic/Algorithms</u>	• 3B-AP-10 • 3B-AP-11 • 3A-AP-12	Elegant solutions are often overlooked or dismissed for naïve ones. Computing the sum of an arithmetic series is a simple constant time algorithm that is used a foundational point for efficient and practical algorithms. Students will survey a variety of topics such as graph theory, compression and more to understand how problem solving and logic can create polynomial time solutions to everyday problems – using Java as a tool to implement these solutions. • Use and adapt classic algorithms to solve computational problems. • Evaluate algorithms in terms of their efficiency, correctness, and clarity. • Design and develop computational artifacts working in team roles using collaborative tools.
<u>Unit 4: Advanced Algorithms/Encryption</u>	• 3A-NI-07 • 2-IC-23	Hot topics in the Computer Science world such as privacy, ethics, cybersecurity, and identity theft are discussed in detail – paying special attention to current events. Students will explore methods of securing data with a focus on encryptions techniques, algorithms, and protocols. While complex math is usually involved – material will be presented that is accessible to all students. Students will then use their Java knowledge to simulate some of the concepts that were covered. • Compare various security measures, considering tradeoffs between the usability and security of a computing system. • Describe tradeoffs between allowing information to be public and keeping information private and secure.
<u>Unit 5: Software/Unit Testing</u>	• 3B-AP-23 • 3B-AP-21 • 2-AP-19 • 2-AP-17	Logic errors are constant plagues to programmers. Compilers' inability to detect these errors demand robust and meticulous testing protocols and procedures to be used by programmers to insure program functionality and lifespan. Key focus will be placed on developing in depth testing routines before any code is ever written. Students will go through a project where they are unable to see or run program code and must develop appropriate testing procedures based on their mental model of what the actual software should do. They will then run their tests against real software and make adjustments/evaluate their success. • Evaluate key qualities of a program through a process such as a code review. • Develop and use a series of test cases to verify that a program performs according to its design specifications. • Document programs in order to make them easier to follow, test, and debug. • Systematically test and refine programs using a range of test cases.
<u>Unit 6: Final Project</u>		Students will have an opportunity to demonstrate all material covered in Computer Science I and II as well as their passion for mathematics in their final project. Working on a tight deadline with numerous constraints, students will experience pressures that software developers deal with in the real world while branching out into intriguing mathematical topics that students have not previously explored.

	• 3A-AP-23 • 3A-AP-13	Java programs will also be created to demonstrate students mathematical research as well as strengthen their arguments. Final presentations of student's findings will take place at the conclusion of the semester. • Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs. • Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
<u>Additional Notes</u>	Students who finish Computer Science I and II and successfully complete Algebra 2 can receive a teacher recommendation to take AP Computer Science A.	