GREAT NECK PUBLIC SCHOOLS
"Where Discovery Leads to Greatness"
#DiscoverGreatness

As always, extra help is in room 123 after school both today and tomorrow.

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 1

Department	Mathematics	
Course Name	Computer Science 1	
Course Length	1 Semester	
High School Credits	0.5	
Description	This course serves as a gateway for high school students to enter the world of Computer Science. The purpose of this course is to build students math and logic skills to serve as a foundation for learning how to read/write algorithms, learn introductory programming constructs to implement said algorithms, and survey current events involving hardware/software. No previous programming experience is required – and the course is designed so that all students can succeed regardless of any prior knowledge. Programming activities will use the Visual Basic programming language.	
Target/eligible students	Grades 9-12, must be enrolled in or have completed Algebra 1	
CSTA Learning Standards Link(s)	Standards | Computer Science Teachers Association (csteachers.org)	
Primary texts and materials	Provided by teacher	
Scope/Sequence	• Overview /Standards	Students will: understand the unique and important differences between Computer Science and programming/coding, learn appropriate and effective problem-solving skills that can be used without technology, work individually and collaboratively, practice communication skills, discover and enhance deductive reasoning, manage a project over a set deadline, practice organization skills, encounter challenging errors and obstacles with the goal of successfully overcoming them.
Unit 1: Intro to Logic (Math/Algebra)	• n/a	This is a fun introductory unit where students get accustomed to working with one another and learning to solve problems on paper and pen before ever getting in front of a computer. Students observe that many difficulty/challenging problems on the surface can be quickly and/or efficiently solved using basic pre-Algebra/Algebra I topics. Focus is spent on having students verbalize problems, ask for clarifications, communicate one on one or in groups, and understanding that the explanation/justifications are almost always more important than the solutions.
Unit 2: Computer Architecture	• 3A-CS-01 • 3A-CS-02 • 2-CS-03 • 3A-CS-03	Students will learn and investigate key technology that exist inside electronics such as CPU, Primary Storage (RAM), Secondary Storage, and peripherals. Students will also get familiar with their computing environment including basic file organization, Google Classroom, cloud storage and basic operating system functions. • Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects • Compare levels of abstraction and interactions between application software, system software, and hardware layers. • Systematically identify and fix problems with computing devices and their components. • Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.

Unit 3: Programming: Basics		Students become familiar with their IDE's (Integrated Development Environments) which include Windows OS, Great Neck Public Schools shared storage, Visual Studio and Google Drive. A workflow is developed to show how project instructions begin to take shape into a tangible program, how all files and deliverables are organized, and how the feedback cycle works with larger scale assignments. Simple instructions and other commands are studied, demonstrated and practiced within the Visual Basic language. Projects may include a simple calculator that can perform addition, and a voting machine that can keep simple tallies.
	• 1A-AP-08 • 2-AP-10 • 2-AP-13 • 1B-AP-15	• Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks. • Use flowcharts and/or pseudocode to address complex problems as algorithms. • Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. • Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.
Unit 4: Programming: Datatypes/Logic		Students will continue to build upon their new foundation in Computer Science by exploring deductive reasoning and how it applies to both everyday life and software. This also includes computing and storing results based off their datatype. Focuses during this unit include how to structure logical arguments that do not involve guessing or making assumptions, logic charts, negations, compound statements, variables, and datatype sizes.
	• 1B-AP-09 • 2-AP-11 • 3B-CS-02	• Create programs that use variables to store and modify data. • Create clearly named variables that represent different data types and perform operations on their values. • Illustrate ways computing systems implement logic, input, and output through hardware components.
Unit 5: Programming: Advanced Design		Students now focus on how interface layout and design factors into software design. Investigating control structures such as timers, visible/nonvisible elements, enabling/disabling elements and more. Students will work on a truncated version of a popular board game such as Yahtzee or Mastermind which will ask as a benchmark up to this point in the course, as well as setting the tone for the upcoming final project.
	• 3A-AP-5 • 1B-AP-10	• Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made. • Create programs that include sequences, events, loops, and conditionals.
Unit 6: Final Project		Students will synthesize all the knowledge covered in the class to create a final project of their choosing. Expectations include demonstrating all topics covered in the course and displaying everything that they have learned into a cohesive and fun educational game. Topics for the game should be academically oriented for grades 9-12. Students will log their work as if they were working for a software company to indicate progress, problems, and setbacks. Presentations of their development process and final product will occur during the final week of the course.
	• 1A-AP-12 • 1B-AP-11 • 2-AP-18	• Develop plans that describe a program's sequence of events, goals, and expected outcomes. • Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process. • Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.

Additional Notes	Students who are passionate about the course are encouraged to take Computer Science II as a follow up class.