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: AP Computer Science A

Department	Mathematics	
Course Name	AP Computer Science A	
Course Length	Full Year	
High School Credits	1	
Description	The main goal of this course is to study the conceptual complexities behind modern programming languages. This includes OOP (Object-Oriented Programming), complex algorithms, choosing appropriate data structures, searching and sorting, and recursion. The Java language is designated by the College Board as the language of study for the course. Students will become comfortable solving challenging problems on paper and in group discussions before attempting to use a computer to implement them. All students take the AP exam in May – which consists of an exam that is 50% multiple-choice exam and 50% writing code on paper.	
Target/eligible students	Grades 11-12, must have completed Computer Science I and II, Algebra 2, and have a recommendation from the teacher.	
Learning Standards Link(s)	AP Computer Science A Course and Exam Description, Effective Fall 2020 (collegeboard.org) pages 33-157	
Primary texts and materials	Provided by teacher	
Scope/Sequence	• Overview /Standards	Students will: Explain how algorithms run-time mirror mathematical functions (such as polynomial, exponential, etc.), discover and utilize primitive datatypes, learn what objects are, how they mirror real world entities and how to use them to create complex programs, demonstrate advanced logic such as truth tables and compound conditional statements, determine when iterative control structures are appropriate and when they are not, what effects they have on the run time of an algorithm/program, when recursive solutions are more elegant, how to write OOP classes according to convention/structure, discover the pros and cons of different data structures and learn how to use them correctly, and discover the complexity of subclasses and inheritance with objects and how they should mirror real world counterparts.
Unit 1: Analysis of Algorithms/Primitive Data		Often students can quickly come up with solutions to problems – but are they efficient? For example – it takes a polynomial amount of time to sum up integers from 1 to N using a loop – but it can take constant time with the use of a non-iterative solution. The first section of the course is used to dispel preconceived notions on solving problems with naïve solutions as opposed to well thought out and more mathematical approaches. Once the tone is set the course transitions into Java basics such as primitive data types, data storage, math/algebra tools, and more essentials skills.
	• VAR-1.B.1	• A type is a set of values (a domain) and a set of operations on them.
	• VAR-1.C.3	• The memory associated with a variable of a primitive type holds an actual primitive value.

	• VAR-1.B.3	• The primitive data types used in this course define the set of operations for numbers and Boolean values.
Unit 2: Object Oriented Programming (Beginning – Using Objects)		Real world entities consist of many properties – a car has a paint color, number of wheels, model year, gas tank capacity, etc. In order to simulate these properties digitally – students will use their knowledge from Unit 1 to save data appropriately and then learn how to structure it in code via OOP or object-oriented programming. The unit will focus on using pre-existing objects first to see how they work and what they are composed of. Students will then solve more complex problems via usage of objects.
	• MOD-1.B.1	• An object is a specific instance of a class with defined attributes.
	• MOD-1.B.2	• A class is the formal implementation, or blueprint, of the attributes and behaviors of an object.
	• MOD-1.C.2	• The parameter list, in the header of a constructor, lists the types of the values that are passed and their variable names. These are often referred to as formal parameters.
	• VAR-1.E.1	• String objects can be created by using string literals or by calling the String class constructor.
	• MOD-1.E.1	• An object's behavior refers to what the object can do (or what can be done to it) and is defined by methods.
	• MOD-1.E.2	• Procedural abstraction allows a programmer to use a method by knowing what the method does even if they do not know how the method was written.
Unit 3: Logic		Program control first relies on appropriate logic. Programmers must be able to take complex situations and not only recreate them digitally – but in a way that the computer can understand and execute. Here we investigate these situations and learn appropriate techniques to write code that is simple to follow while simultaneously being efficient. We investigate classic laws of logic that can be applied to our code to help with redundant statements and poor logic.
	• CON-2.A.1	• Conditional statements interrupt the sequential execution of statements.
	• CON-2.A.3	• A one-way selection (if statement) is written when there is a set of statements to execute under a certain condition. In this case, the body is executed only when the Boolean condition is true.
	• CON-2.A.5	• A multi-way selection is written when there are a series of conditions with different statements for each condition. Multi-way selection is performed using if-else-if statements such that exactly one section of code is executed based on the first condition that evaluates to true.
	• CON-1.G.1	• De Morgan's Laws can be applied to Boolean expressions.
	• CON-1.G.2	• Truth tables can be used to prove Boolean identities.

Unit 4: Iteration and Recursion		Control structures such as loops, and recursion allow for repetition when appropriate. Students have already had tons of practice at this point solving complex problems without access to these tools – and they are now rewarded by learning about them and adding them to their toolboxes for future use on problems. We investigate classical situations that require loops – like an input system for entering a variable amount of data, warehouse databases, etc. Students learn the different constructs available in Java, as well as how to analyze the run times of both iterative and recursive statements on paper before implementing them in code.
	• CON-2.C.1	• Iteration statements change the flow of control by repeating a set of statements zero or more times until a condition is met.
	• CON-2.H.1	• A statement execution count indicates the number of times a statement is executed by the program.
	• CON-2.E.5	• "Off by one" errors occur when the iteration statement loops one time too many or one time too few.
	• CON-2.C.2	• In loops, the Boolean expression is evaluated before each iteration of the loop body, including the first. When the expression evaluates to true, the loop body is executed. This continues until the expression evaluates to false, whereupon the iteration ceases.
	• CON-2.O.2	• Recursive methods contain at least one base case, which halts the recursion, and at least one recursive call.
	• CON-2.O.5	• Any recursive solution can be replicated through the use of an iterative approach.
	• CON-2.P.4	• The binary search algorithm can be written either iteratively or recursively.
Unit 5: Object Oriented Programming (Intermediate – Writing Classes)		Building upon Unit 2 – now that students are familiar with using objects – they will begin to construct them from scratch. This includes coming up with appropriate constructors, attributes, and accessors/mutators. These objects will often be digital representations of real objects. One project involves students recreating the game Scrabble – where each tile, the board, the player, etc. all must be represented by separate objects that all need to work together to simulate the actual game.
	• MOD-3.A.1	• Data encapsulation is a technique in which the implementation details of a class are kept hidden from the user.
	• MOD-2.B.1	• An object's state refers to its attributes and their values at a given time and is defined by instance variables belonging to the object. This creates a "has-a" relationship between the object and its instance variables.
	• MOD-2.C.5	• Programmers write method code to satisfy the postconditions when preconditions are met.
	• MOD-2.D.1	• An accessor method allows other objects to obtain the value of instance variables or static variables.
	• VAR-1.G.4	• Through method decomposition, a programmer breaks down a large problem into smaller subproblems by creating methods to solve each individual subproblem.

Unit 6: Data Structures		Variables are important in both monolithic programs as well as object-oriented ones. However – individual storage of each piece of data certainly is not scalable in the long run. Similar to how students were rewarded with the use of iterative structures to solve problems they learned to solve without them – the same reward is applied here with respect to data structures. Students will gain access to arrays in one and two dimensions, as well as an object-oriented data structure called an ArrayList that allows them to hold large amounts of data in a single variable. Sorting and searching algorithms now come into play and unlock new powerful methods to use in problem solving.
	• VAR-2.A.1	• The use of array objects allows multiple related items to be represented using a single variable.
	• VAR-2.B.1	• Iteration statements can be used to access all the elements in an array. This is called traversing the array.
	• CON-2.I.1	• There are standard algorithms that utilize array traversals to: o Determine a minimum or maximum value o Compute a sum, average, or mode o Determine if at least one element has a particular property o Determine if all elements have a particular property o Access all consecutive pairs of elements o Determine the presence or absence of duplicate elements o Determine the number of elements meeting specific criteria
	• VAR-2.D.1	• An ArrayList object is mutable and contains object references.
	• CON-2.J.1	• There are standard ArrayList algorithms that utilize traversals to: o Insert elements o Delete elements o Apply the same standard algorithms that are used with 1D arrays
	• CON-2.K.2	• Sequential/linear search algorithms check each element in order until the desired value is found or all elements in the array or ArrayList have been checked.
	• CON-2.M.1	• Informal run-time comparisons of program code segments can be made using statement execution counts.
Unit 7: Object Oriented Programming (Advanced – Writing Subclasses/Inheritance)		The final stage required to master OOP is the conceptual understanding of subclasses. For example, in Geometry we have many shapes such as parallelograms and squares – where a square has its own unique properties as well as all of those of a parallelogram. If we were to code these (instead of doing it as two completely separate files) we can instead minimally write code for square and have it inherit all object properties from parallelogram. This allows students to create even more complex systems than before – while eliminating redundancy and repetitive code.
	• MOD-3.B.1	• A class hierarchy can be developed by putting common attributes and behaviors of related classes into a single class called a superclass.
	• MOD-3.B.12	• A subclass is usually designed to have modified (overridden) or additional methods or instance variables.
	• MOD-3.B.13	• A subclass will inherit all public methods from the superclass; these methods remain public in the subclass.

	• MOD-3.D.2 • MOD-3.B.2	• At compile time, methods in or inherited by the declared type determine the correctness of a non-static method call. • Classes that extend a superclass, called subclasses, can draw upon the existing attributes and behaviors of the superclass without repeating these in the code.
Unit 8: AP Exam Review/Post Exam	n/a	Students will spend several weeks reviewing material to prepare for the AP Exam. This exam has two equally weighted components – multiple choice and free response. After the exam has concluded, we will explore many challenging topics such as Dynamic Programming, discrete math, and other topics of the students choosing.
Additional Notes	Topics order subject to change based on class needs. Students who are passionate about the course are encouraged to take AP Computer Science Principles if they have not done so already.	