Subject:	Computer Science	**Course/Grade Level:**	Programming Challenges / 11th-12th
Focus Statement:	Students will compete in programming competitions and utilize common algorithms to solve the competition challenges.		

Outcome 1:

CTE.PROG.1		**Students will select a programming language to use in their programming challenges.**	
Pacing:		**Local Code:**	**Components:**
Instruct	**Assess**		**Students will:**
NA	NA	CTE.PROG.1.1	Choose from C, C++, and Java to utilize during programming challenges.
NA	NA	CTE.PROG.1.2	Explain their rationale for their programming language of choice.
NA	NA	CTE.PROG.1.3	Utilize the standard input/output of their programming language of choice.
NA	NA	CTE.PROG.1.4	Write comments in their programming language of choice.
NA	NA	CTE.PROG.1.5	Create constant variables in their programming language of choice.
NA	NA	CTE.PROG.1.6	Create functions/methods/subroutines in their programming language of choice.
NA	NA	CTE.PROG.1.7	Utilize arrays in their chosen programming language.
NA	NA	CTE.PROG.1.8	Utilize two-dimensional arrays in their chosen programming language.
NA	NA	CTE.PROG.1.9	Utilize sentinels to guard against arrays out-of-bounds errors.
NA	NA	CTE.PROG.1.10	Create records/structures in their chosen programming language.

Outcome 2:

CTE.PROG.2		**Students will utilize elementary data structures in their chosen programming language.**	
Pacing:		**Local Code:**	**Components:**
Instruct	**Assess**		**Students will:**
NA	NA	CTE.PROG.2.1	Utilize stacks in their chosen programming language.
NA	NA	CTE.PROG.2.2	Utilize queues in their chosen programming language.
NA	NA	CTE.PROG.2.3	Implement dictionaries in their chosen programming language.
NA	NA	CTE.PROG.2.4	Implement a priority queue in their chosen programming language.
NA	NA	CTE.PROG.2.5	Implement sets in their chosen programming language.
NA	NA	CTE.PROG.2.6	Understand how their chosen programming language implements strings.
NA	NA	CTE.PROG.2.7	Know some helpful string methods in their chosen programming language.
NA	NA	CTE.PROG.2.8	Know the nine situations where sorting can be used to solve programming challenges.
NA	NA	CTE.PROG.2.9	Utilize built-in sorting algorithms in their chosen programming language.

Outcome 3:

CTE.PROG.3		**Students will utilize arithmetic algorithms to solve programming challenges.**	
Pacing:		**Local Code:**	**Components:**
Instruct	**Assess**		**Students will:**
NA	NA	CTE.PROG.3.1	Utilize the built-in mathematical methods in their chosen programming language.
NA	NA	CTE.PROG.3.2	Implement large integers using arrays of digits.
NA	NA	CTE.PROG.3.3	Implement addition for large integers.

NA	NA	CTE.PROG.3.4	Implement subtraction for large integers.
NA	NA	CTE.PROG.3.5	Implement multiplication for large integers.
NA	NA	CTE.PROG.3.6	Implement division for large integers.
NA	NA	CTE.PROG.3.7	Implement comparison operations for large integers.
NA	NA	CTE.PROG.3.8	Demonstrate how to compare real numbers.
NA	NA	CTE.PROG.3.9	Manipulate polynomials in their chosen programming language.
NA	NA	CTE.PROG.3.10	Determine if the product rule applies to a given programming challenge problem.
NA	NA	CTE.PROG.3.11	Determine if the sum rule applies to a given programming challenge problem.
NA	NA	CTE.PROG.3.12	Determine if the inclusion-exclusion formula applies to a given programming challenge problem.
NA	NA	CTE.PROG.3.13	Determine if a recurrence relation applies to a given programming challenge problem.
NA	NA	CTE.PROG.3.14	Determine if a binomial coefficient applies to a given programming challenge problem.
NA	NA	CTE.PROG.3.15	Solve a programming challenge problem using recursion and induction.
NA	NA	CTE.PROG.3.16	Write an algorithm to determine if a given number is prime.
NA	NA	CTE.PROG.3.17	Write an algorithm to determine if a given number is divisible by another given number.
NA	NA	CTE.PROG.3.18	Write an algorithm to find the greatest common divisor of a given number.
NA	NA	CTE.PROG.3.19	Write an algorithm to find the least common multiple of a given number.
NA	NA	CTE.PROG.3.20	Use modulus to solve programming challenge problems.

Outcome 4:

CTE.PROG.4		**Students will utilize backtracking algorithms to solve problems with a large search space.**	
Pacing:		**Local Code:**	**Components:**
Instruct	**Assess**		**Students will:**
NA	NA	CTE.PROG.4.1	Write a recursive backtracking algorithm.
NA	NA	CTE.PROG.4.2	Use a pruning search to remove candidates from a possible solution set.
NA	NA	CTE.PROG.4.3	Solve a programming challenge problem using a backtracking algorithm.

Outcome 5:

CTE.PROG.5		**Students will utilize graphs to solve programming challenge problems.**	
Pacing:		**Local Code:**	**Components:**
Instruct	**Assess**		**Students will:**
NA	NA	CTE.PROG.5.1	Demonstrate the difference between undirected and directed graphs.
NA	NA	CTE.PROG.5.2	Demonstrate the difference between weighted and unweighted graphs.
NA	NA	CTE.PROG.5.3	Demonstrate the difference between cyclic and acyclic graphs.
NA	NA	CTE.PROG.5.4	Demonstrate the difference between simple and non-simple graphs.
NA	NA	CTE.PROG.5.5	Demonstrate the difference between embedded and topological graphs.
NA	NA	CTE.PROG.5.6	Demonstrate the difference between implicit and explicit graphs.
NA	NA	CTE.PROG.5.7	Demonstrate the difference between labeled and unlabeled graphs.
NA	NA	CTE.PROG.5.8	Determine a data structure to represent a graph when solving a programming challenge problem.
NA	NA	CTE.PROG.5.9	Traverse a graph using a breadth-first or depth-first search.

NA	NA	CTE.PROG.5.10	Utilize topological sorting to solve problems involving directed acyclic graphs.
NA	NA	CTE.PROG.5.11	Understand the basic principles of graph theory.
NA	NA	CTE.PROG.5.12	Generate a minimum spanning tree using Prim’s algorithm.
NA	NA	CTE.PROG.5.13	Utilize Dijkstra’s algorithm to find the shortest path in a weighted graph.
NA	NA	CTE.PROG.5.14	Utilize Floyd’s all-pairs shortest-path algorithm to length of the shortest path between all pairs of vertices in a given graph.
NA	NA	CTE.PROG.5.15	Write an algorithm to solve network flow problems.