WASHINGTON
LEADERSHIP
ACADEMY

Advanced Math I:
functions and modeling in computing

Bootstrap

+ computing creatively
thriving mathematically

(A course adapted from and made in partnership with Bootstrap)

Washingtonleadershipacademy.org

Course Overview for Advanced Math I

Bootstrap is a 20-25 hour curriculum for students aged 12 – 16. Advanced Math I is an implementation of Bootstrap (bootstrapworld.org) lasting the duration of one interim session, which teaches students to program their own videogames using a special, algebraic programming language. By learning a unique process to break down and solve word problems (many taken from state standardized tests), students work with order of operations, linear and piecewise functions, inequalities in the plane, and the Pythagorean theorem in order to build a game of their own design. Using a process called the "Design Recipe", students must think through word problems step-by-step on paper, declaring the domain and range of each function, completing a series of worked examples, and identifying their variables before writing their function definition. As the main project in this course, students create a simple, 3-character game involving a player, a target and a danger. They design what each character looks like, and use algebraic concepts to detect collisions, handle keystrokes, and determine how they move and interact.

Common Core Alignment Advanced Math I is aligned to Common Core Standards for Mathematics, covering most Functional and Algebraic standards across the Common Core from Grade 7 through Algebra 2. This includes Mathematical Practice standards as well as Mathematical Content standards. Additionally, Advanced Math I is aligned to the CSTA Computer Science standards. This alignment makes it possible and flexible for teachers to integrate the course into the classroom, allowing teachers to use time already planned into their pacing guidelines or scope and sequence plans. Because Bootstrap (the course on which Advanced Math I and II are based) is meant to be taught as part of an algebra class, it ensures that every algebra student will be exposed to an innovative, engaging, and effective way of teaching computer programming and algebra which will build their confidence and abilities, and stay with them in other math courses they pursue.

Core Competencies:

Computational Thinking Practices

- Analyze the effects of the developments in computing
- Design and implement creative solutions and artifacts
- Apply abstractions and models
- Analyze your own computational work and the work of others
- Communicate computational thought processes, procedures, and results to others
- Collaborate with peers on computing activities

Standards of Math Practice

- Make sense of problems and persevere in solving them
- Reason abstractly and quantitatively
- Construct viable arguments and critique the reasoning of others
- Model with mathematics
- Use appropriate tools strategically
- Attend to precision
- Look for and make use of structure
- Look for and express regularity in repeated reasoning

Modules Overview

- *Module 1: Videogames and Coordinate Planes*
- *Module 2: Contracts, Strings, and Images*
- *Module 3: Introduction to Definitions*
- *Module 4: Manipulating Images and Making Flags*
- *Module 5: The Design Recipe*
- *Module 6: Game Animation*
- *Module 7: Teaching Functions to Compare*
- *Module 8: Conditional Branching*
- *Module 9: Piecewise Practice and 2D Movement*
- *Module 10: Collision Detection*
- *Module 11: Prepping for Launch*

Module Assessments

A Common Timeframe for Modules and Interim Assessments The Advanced Math curriculum will utilize performance-based assessments after each module to enhance and reinforce learning, while giving students a project in which to apply the algebra and programming skills learned in each module. Group-based performance tasks will also be given at regular intervals to reinforce the teamwork and communication skills embedded into the curriculum. These tasks will align themselves to the PARCC Performance Level Descriptors.

Instructional Model and Implementation Strategies for Advanced Math I

Students will take Advanced Math I every other day for 90-minutes for the duration of one interim. The course is intended to thread together Algebraic concepts into computing. Using a linear design sequence students develop functional programming and coding language skills using mathematical concepts. As students progress through nodes within a module they collectively piece together the elements of a single project outcome, the design of a personally crafted video game. Students will arrive at this outcome using the following instructional strategies:

Accountable video-based Instruction: videos used to introduce or review a specific scientific concept that includes practice problems and scenarios within the video ensuring that students attention and time spent. Student usage data is tracked as well as their achievement data on video-associated assessments.

Design Nodes: Students engage in a stepwise design process to achieve a large outcome. Students work through smaller outcomes (nodes) that when combined or sequenced together make the large goal of the process occur.

Skills-based Workshops: Students work through a self-paced skills workshop designed to lead students to acquire a specific skill. Workshops typically use a gamified or challenge-styled approach to building deepening levels of complexity around a skill.

Engineering and Design Thinking Challenges: students are provided with a challenge to program or code a solution, test prototypes, and design new models around a computing outcome. Students work independently and collaboratively to reach specific outcomes.

Standards Coverage

Interim Session	Date Range	Standards Covered
As Applies	**As Applies** (47 days)	***CCSS*** N-Q, 6.NS.5-8, 7-EE.1-4, 8.F.1-3, A-SSE.1-2, A-SSE.3-4, A-CED.1-4, A-REI.1-2, A-REI.3-4, A-REI.10-12, 8.G.6-8, F-IF.1-3, F-IF.4-6, F-IF.7-9, F-BF.1-2, F-BF.3-4, F-LE.1-4, F-LE.5, F-TF.5 ***SMPs*** MP.1-8 ***CSTA Standards*** L1:6:CT.1, L1:6:CPP.6, L2:CT:1, L2:CT:6, L2:CT:7, L2:CT:14, L2:CPP:4, L2:CPP:5, L2:CPP:9, L3:CT:A2, L3:CPP:A2, L3:CPP:A3, L3:CPP:A4, L3:CPP:A8, L3:CPP:A12, L3:CT:B5, L3:CT:B6, L3:CT:B10, L3:CPP:B2, L3:CPP:B7

Module #1: Videogames and Coordinate Planes

In this module students discuss the components of their favorite videogames, and discover that they can be reduced to a series of coordinates. They then explore coordinates in Cartesian space, and identify the coordinates for the characters in a game at various points in time. Once they are comfortable with coordinates, they brainstorm their own games and create sample coordinate lists for different points in time in their own game. Students begin by creating a data model that describes a simple videogame, and identify the coordinates of characters in a picture of that game. After brainstorming their own videogame, students use the order of operations to convert several arithmetic expressions between multiple representations.

# of Days	Standards	Course Resources	Assessments
2	5.G.1-2, 5.OA.1-2, 6.NS.5-8, A-SSE.1-2, A-SSE.3-4, N-Q, A-REI.1-2	**Bootstrap:1 Unit 1** **http://www.bootstrapworld.org/materials/spring2016/courses/bs1/units/unit1/index.html**	**Bootstrap PBA I and Performance Tasks**

Module #2: Contracts, Strings, and Images

Students are introduced to a set-mapping representation for functions, in which the function object exists as a means of translating points from a Domain into a Range. Coupled with their understanding of Circles of Evaluation, students generalize their understanding of functions to include other datatypes, including Strings and Images. As part of this module, students will enter (evaluate) expressions for generating Strings and Images. They will also write down Contracts (name, domain, and range) for arithmetic expressions, as well as several image-producing expressions.

# of Days	Standards	Course Resources	Assessments
2	A-SSE.1-2, F-IF.1-3, N-Q, A-REI.1-2	**Bootstrap:1 Unit 2** **http://www.bootstrapworld.org/materials/spring2016/courses/bs1/units/unit2/index.html**	**Bootstrap PBA 2 and Performance Tasks**

Module #3: Introduction to Definitions

Students are introduced to the Definitions area in the WeScheme IDE, and learn the syntax for defining values of various types. They will name their videogame project, and modify pre-written definitions for their title and character images. They are also introduced to the syntax of defining functions and creating examples, and will define at least two functions based on word problems using the Design Recipe.

# of Days	Standards	Course Resources	Assessments
2	6.NS.5-8, 7.EE.1-4, A-SSE.1-2, F-BF.1-2, F-IF.1-3, F-IF.4-6, F.IF.7-9, N-Q, F-LE.1-4	**Bootstrap:1 Unit 3** **http://www.bootstrapworld.org/materials/spring2016/courses/bs1/units/unit3/index.html**	**Bootstrap PBA 3 and Performance Tasks**

Module #4: Manipulating Images and Making Flags

Students create scaled, rotated, flipped, and layered images. They use basic value definitions of shapes and their own created images to produce images for various nations' flags.

# of Days	Standards	Course Resources	Assessments
2	A-SSE.1-2, F-IF.1-3, F-IF.4-6, F.IF.7-9, F-LE.5	**Bootstrap:1 Supplemental** **http://www.bootstrapworld.org/materials/spring2016/courses/bs1/units/Supplemental/index.html**	**Bootstrap PBA 4 and Performance Tasks**

Module #5: The Design Recipe

Students continue to practice the Design Recipe by applying it to simple problems.They will use the Design Recipe to define a function used to make a rocket fly, and write functions to solve other simple word problems by using the Design Recipe

# of Days	Standards	Course Resources	Assessments
2	8.F.1-3, A-SSE.1-2, F-BF.1-2, F-BF.3-4, F-IF.1-3, F.IF.4-6, F.IF.7-9	**Bootstrap:1 Unit 4** **http://www.bootstrapworld.org/materials/spring2016/courses/bs1/units/unit4/index.html**	**Bootstrap PBA 5 and Performance Tasks**

Module #6: Game Animation

Students define functions that map attributes of their game from one frame to the next, allowing them to move their dangers, targets, and projectiles. They will write functions to add danger and target movement to their games.

# of Days	Standards	Course Resources	Assessments
2	F-IF.1-3, F-LE.5, F-BF.1-2, F.IF.7-9, N-Q	**Bootstrap:1 Unit 5** **http://www.bootstrapworld.org/materials/spring2016/courses/bs1/units/unit5/index.html**	**Bootstrap PBA 6 and Performance Tasks**

Module #7: Teaching Functions to Compare

Students discover Boolean types, and use them to create programs that test values, and then model scenarios using these programs. Students detect when game elements have moved offscreen (so they can reappear on the other edge of the screen)

# of Days	Standards	Course Resources	Assessments
2	7.EE.1-4, 8.F.1-3, A-CED.1-4, A-SSE.1-2, F-IF.1-3, F-IF.4-6, F-IF.7-9, F-BF.1-2, F-BF.3-4, A-REI.3-4, A-REI.10-12	**Bootstrap:1 Unit 6** **http://www.bootstrapworld.org/materials/spring2016/courses/bs1/units/unit6/index.html**	**Bootstrap PBA 7 and Performance Tasks**

Module #8: Conditional Branching

Students learn about and practice writing piecewise functions using the design recipe. They will write update-player, a piecewise function which moves their player in response to key-presses.

# of Days	Standards	Course Resources	Assessments
2	A-SSE.1-2, A-CED.1-4, F-BF.1-2, F-BF.3-4, F.IF.7-9	**Bootstrap:1 Unit 7** **http://www.bootstrapworld.org/materials/spring2016/courses/bs1/units/unit7/index.html**	**Bootstrap PBA 8 and Performance Tasks**

Module #9: Piecewise Practice and 2D Movement

Students practice writing more piecewise functions based on word problems, and extend their update-player function to include "cheat codes", boundary detection, and 2-dimensional movement using data structures to alter both the character's x and y-coordinates.

# of Days	Standards	Course Resources	Assessments
2	A-SSE.1-2, F-IF.1-3, F-IF.4-6, F.IF.7-9, F-LE.5	**Bootstrap:1 Supplemental** **http://www.bootstrapworld.org/materials/spring2016/courses/bs1/units/Supplemental/index.html**	**Bootstrap PBA 9 and Performance Tasks**

Module #10: Collision Detection

Students derive, discuss, and prove the Pythagorean theorem, then use this theorem—in conjunction with Booleans—to detect collisions in their games. Students will write the distance function in code into their game files, and add a collide? function to their games in order to detect when the player has collided with another game character.

# of Days	Standards	Course Resources	Assessments
2	6.NS.5-8, 8.F.1-3, 8.G.6-8, A-SSE.1-2, F-BF.1-2, F-BF.3-4, F-IF.1-3, F-IF.4-6, F.IF.7-9	**Bootstrap:1 Unit 8** **http://www.bootstrapworld.org/materials/spring2016/courses/bs1/units/unit8/index.html**	**Bootstrap PBA 10 and Performance Tasks**

Module #11: Prepping for Launch

Students edit details of their games, practice talking about their code, creating a poster illustrating a key concept in their game program, and prepare for a videogame launch party!

# of Days	Standards	Course Resources	Assessments
2	F-BF.1-2, F.IF.7-9	**Bootstrap:1 Unit 9** **http://www.bootstrapworld.org/materials/spring2016/courses/bs1/units/unit9/index.html**	**Bootstrap PBA 11 and Performance Tasks**