PLANNED INSTRUCTION

A PLANNED COURSE FOR:

Computer Programming 1

____

Curriculum writing committee:
Jessica Hubal

Grade Level: 9-12

Date of Board Approval: ______2021____________

Time/Credit for the Course: Half Year, .5 CREDIT, 90 days, meeting 1 period per day

Computer Programming 1

Programs	75%
Quizzes	20%
Participation	5%

Curriculum Map

1. **Marking Period One:**

    - **Overview based on 45 days:**

      Unit 1: Basic Components (Events, Labels, Buttons, Variables, Radio Buttons)
        - Day 1: Reflect upon the importance of coding and programming languages.
        - Days 2-6: Represent a step-by-step algorithmic progress using sequential code statements. Introduce App Lab features (Code.org's JavaScript programming environment) and investigate events, labels, buttons, picture boxes and radio buttons.
        - Days 7-15: Incorporate variables of different data types into applications, specifically integers, doubles, strings and Booleans. Evaluate expressions that use arithmetic and relational operators and manipulate strings (concatenation).

      Unit 2: Decision Structures
        - Days 16-45: Write conditional statements, as well as nested conditional statements, to determine result. Construct and use counter/accumulator variables within structures. Write and evaluate expressions that use logical operators.
    - **Goals:**
        - Build a graphical user interface on applications that include multiple components, such as labels, buttons, radio buttons, user input (text boxes/prompts), picture boxes, etc.
        - Create programs that declare and initialize variables.
        - Declare variables identifying the correct data type.
        - Identify the correct graphical user interface components for a given problem.
        - Build programs that utilize the screen and event procedures properly.

- Develop and implement algorithms that incorporate decision structures (Conditionals -If's, If-Else's, Nested If- Structures).
  - Construct and use counter and accumulator variables.
  - Determine the results of code segments.
- **Big Ideas:**
  - Programming enables problem solving, human expression, and creation of knowledge.
  - Algorithms are used to develop and express solutions to computational problems.
  - Programmers can use a formal, iterative design process or a less rigid process of experimentation.
  - Programmers will encounter phases of investigating and reflecting, designing, prototyping, and testing.

2. **Marking Period Two:**
   - **Overview based on 45 days:**

     Unit 3: Control Structures
     - Days 1-8: Incorporate iteration (both while and for loops) and selection into code as a way of providing instructions for the compiler to process each of the many possible input values.
     - Days 9-12: Develop iteration conditions that utilize sentinel (flag) values.
     - Days 12-15: Evaluate expressions that use modulus operators and utilize them to determine if input/random values are even or odd.

     Unit 4: Functions
     - Days 16-25: Break down program code into smaller, more manageable pieces by creating functions. Write statements to call functions and determine the result or effect of a function call. Develop functions that utilize arguments, parameters and return values.

     Unit 5: String Functions
     - Days 26-35: Write and evaluate expressions that manipulate strings using pre-made String functions (substring, replace, index Of, etc.). Incorporate iteration to manipulate strings to solve problems.

     Unit 6: Final Application and Poster
     - Days 36-45: Develop a computer program of student choice that incorporates key elements learned within the course. Create a poster that directly states the title, goals(s) and purpose of program, as well as displays code/application screen(s) and direct project link using a Quick Response (QR) code.

- **Goals:**
    - Identify problems where loops and counter/accumulating variables need to be utilized.
    - Create programs that use iteration (while, for) and sentinels (flags) conditions.
    - Create programs that utilize counter and accumulator variables, as well as modulus operators to produce more efficient code.
    - Build applications that use the appropriate String functions to complete the specified task.
    - Recognize situations where decision and control structures should be used to make more efficient use of the code.
- **Big Ideas:**
    - Programmers integrate algorithms and abstraction to create programs for creative purposes and to solve problems.
    - Programmers need to think algorithmically and use abstraction to define and interpret processes that are used in a program.
    - Programmers can use a formal, iterative design process or a less rigid process of experimentation.
    - Programmers will encounter phases of investigating and reflecting, designing, prototyping, and testing.

Textbook and Supplemental Resources:

- Code.org’s App Lab Built-in Application Programming Interface
    - https://studio.code.org/docs/applab/
- Rauschmayer, A. (2014). Speaking JavaScript: An in-depth guide for programmers. Sebastopol, CA: O'Reilly Media.
    - Free Online Text Available at http://speakingjs.com/es5/index.html
- CodeHS.com’s Introduction to Computer Science in JavaScript (Bulldog) course
- Code.org’s CS Fundamentals Express course
- Code.org’s CS Discoveries course: Unit 3 – Interactive Animations and Games

Curriculum Plan

Unit 1: Basic Components (Events, Labels, Buttons, Variables, etc.) 15 days

Standards:

- *Pennsylvania Department of Education Computer Science Standards*
  - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
  - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
  - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
  - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
  - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
  - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
  - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
  - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
  - 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
  - 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
  - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
  - 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

Objectives:

- Apply the process used for creating programs. (DOK-4)
- Recognize, label, and place text boxes, labels, buttons, and radio buttons on App Lab application. (DOK -1)
- Construct an application. (DOK-2)
- Use operators and expressions. (DOK-1)
- Represent a value with a variable. (DOK-2)
- Create variable assignments. (DOK-4)

- Use rounding functions. (DOK-1)
- Determine the value of a variable as a result of an assignment. (DOK-3)
- Classify appropriate data types. (DOK-2)
- Construct multiple variables in programming assignments. (DOK-3)
- Evaluate expressions that use arithmetic operators and relational operators (DOK-2)
- Create event procedures. (DOK-2)
- Design App Lab applications that solve problems. (DOK-4)
- Implement and apply an algorithm. (DOK-4)

Core Activities and Corresponding Instructional Methods:

- Place App Lab components on an application and rename the properties.
    - Direct instruction and practice on App Lab components using SMART technology (i.e., the toolbox, the properties window, building screen, etc.).
    - Teacher led demonstration using different application examples.
    - Lead a classroom discussion that prompts students to compare and contrast App Lab components and their purposes.
    - Other possible strategies: code tracing, error analysis, work backwards.
- Write the “Hello, World” application.
    - Classroom discussion and guided practice on building graphical user interfaces for program development.
    - Other possible strategies: code tracing, error analysis, work backwards.
- Write a program that allows the students to practice using buttons, textboxes, and labels. The program will also incorporate mathematical equations to arrive at a solution. Variable assignment to the appropriate data type will also be exhibited by the student in this program.
    - Direct instruction and practice on App Lab components (i.e., calculating the mean of user input, incorporating operators, symbols, data types, and variable declaration statements, and using proper programming design practices.
    - Use diagnostic assessment and questioning to evaluate students’ knowledge of proper data types and their uses.
    - Teacher led demonstration using different programming controls in example code.
    - Classroom discussion and guided practice on building graphical user interfaces with different components for program development.
    - Other possible strategies: code tracing, error analysis, work backwards.
- Write a program with option buttons that allows the user to choose between different mathematical operations. The program will propose a problem that requires students to create solutions using logical reasoning.

- Teacher led demonstration on how to write a program using option buttons and checkboxes.
- Direct instruction and practice on mathematical operations and their operators, as well as App Lab components and their event procedures.
- Other possible strategies: code tracing, error analysis, work backwards.

Assessments:

- **Diagnostic:** Algebra 1 Common Assessment / Keystone Algebra 1 Test Results
- **Formative:**
    - Questioning and analysis of student work (e.g., classroom assignments, work samples, etc.).
    - Unit 1 Quiz
    - Practice programming assignments
- **Summative:**
    - Graded programming assignments
    - Unit 1 Quiz

Unit 2: Decision Structures 30 days

Standards:

- *Pennsylvania Department of Education Computer Science Standards*
    - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
    - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
    - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
    - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
    - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
    - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
    - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
    - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
    - 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
    - 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
    - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
    - 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
    - 3B-AP-14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

Objectives:

- Use and appropriately label message boxes, check boxes, counter variables and accumulating variables in programs. (DOK-1)
- Generate and use random numbers. (DOK-1)
- Design App Lab applications that solve problems. (DOK-4)

- Create decision structures to develop algorithms and control the flow of execution in a App Lab program. (DOK-4)
- Determine the result of conditional statements. (DOK-2)
- Write nested conditional statements and determine the result of nested conditional statements. (DOK-4)
- Design Boolean expressions using appropriate logical operators to test conditions and decision structures. (DOK-4)
- Apply the concept of good program design and good programming style in applications. (DOK-3)

Core Activities and Corresponding Instructional Methods:

- Write programs that utilize decision structures (including nested), such as a guessing game or a payroll application.
    - Lead a classroom discussion that prompts students to create accurate conditions and given different problem statements.
    - Teacher led demonstration on how to incorporate if, if-else, and if-else-if statements to complete given tasks.
    - Direct instruction and practice on App Lab components, data types, if-statements, conditions, and proper documentation practices.
- Write a program that receives input from textboxes that calculates the addition, subtraction, multiplication, or division of two numbers, depending on which option button the user clicks on.
    - Direct instruction and practice on App Lab components, data types, if-statements, conditions, event procedures, and proper documentation practices.
    - Guided practice: Students will write programs that use option buttons, checkboxes, and if, if-else, and if-else-if statements to complete a given task.
    - Classroom discussion and guided practice on creating programs using different forms of the if-statement.
- Write programs that utilize compound Boolean expressions within a decision structure conditional, such as a test grade checker or a rock, paper, scissors application.
    - Lead a classroom discussion that prompts students to create accurate conditions and given different problem statements.
    - Teacher led demonstration on how to incorporate if, if-else, and if-else-if statements to complete given tasks.
    - Direct instruction and practice on App Lab components, data types, if-statements, conditions, and proper documentation practices.
- Write programs that utilize counting variables, such as a pizza order, a sandwich order, and a modified blackjack card game.

- Lead a classroom discussion that prompts students to create accurate conditions and given different problem statements.
- Teacher led demonstration on how to incorporate if, if-else, and if-else-if statements to complete given tasks.
- Direct instruction and practice on App Lab components, data types, if-statements, conditions, and proper documentation practices.

Assessments:

- **Diagnostic:** Analysis of student work (e.g., classroom assignments, work samples, quizzes)/Observation and anecdotal notes
- **Formative:**
    - Diagnostic assessment and questioning
    - Unit 2 Quiz
    - Practice programming assignments
- **Summative:**
    - Graded programming assignments
    - Unit 2 Quiz

Unit 3: Control Structures 15 days

Standards:

- *Pennsylvania Department of Education Computer Science Standards*
  - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
  - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
  - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
  - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
  - 3B.AP.14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
  - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
  - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
  - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
  - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
  - 1B-AP-08: Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
  - 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
  - 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
  - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
  - 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
  - 3B-AP-14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
  - 3A-AP-15: Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.

- 3A-AP-16: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instruction.
- 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

Objectives:

- Express an algorithm that uses iteration without using a programming language. (DOK-2)
- Design algorithms and write programs that use looping structures. (DOK-4)
- Determine the result or side effect of iteration statements. (DOK-3)
- Differentiate between event controlled loops and count controlled loops. (DOK-3)
- Apply the concepts of sentinels (flags) and events to control loops. (DOK-4)
- Use counter and accumulator variables. (DOK-1)
- Design program with the while and for loops. (DOK-4)
- Compare multiple algorithms to determine if they yield the same side effect or result. (DOK-3)
- Apply the concept of good program design and good programming style in applications. (DOK-3)

Core Activities and Corresponding Instructional Methods:

- Write programs that use “while” and “for” loops.
  - Teacher led demonstration on building programs that need loops for efficiency.
  - Guided practice: Include step-by-step written explanation of how to create a complex program that makes use of loops.
  - Direct instruction and practice on programs that utilize iteration

  Programs include:
    - Adding a set of numbers together and display the sum in a label
    - Finding the factorial of a number
    - Finding the sum of odd numbers (numbers 1 to a maximum number entered)
    - Finding the average score of entered test scores and bowling scores
    - Generating a unique random number and finding the number of iterations it took

Assessments:

- **Diagnostic:** Analysis of student work (e.g., classroom assignments, work samples, quizzes)/Observation and anecdotal notes
- **Formative:**
  - Diagnostic assessment and questioning
  - Unit 3 Quiz
  - Practice programming assignments
- **Summative:**
  - Graded programming assignments
  - Unit 3 Quiz

Unit 4: Functions 10 days

<u>Standards:</u>

- *Pennsylvania Department of Education Computer Science Standards*
    - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
    - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
    - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
    - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
    - 3B.AP.14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
    - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
    - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
    - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
    - 1B-AP-08: Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
    - 1B-AP-11: Decompose (break down) problems into smaller, manageable sub problems to facilitate the program development process.
    - 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
    - 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
    - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
    - 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
    - 3B-AP-14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.

    - 3A-AP-15: Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.
    - 3A-AP-16: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instruction.
    - 3A-AP-17: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
    - 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

Objectives:

- Write statements to call functions. (DOK-1)
- Define function calls, arguments, parameters, and function returns. (DOK-1)
- Determine the result or effect of a function call. (DOK-3)
- Analyze how the use of function abstraction manages complexity in a program. (DOK-4)
- Develop procedural abstractions to manage complexity in a program by writing functions. (DOK-4)
- Modify call functions to incorporate arguments. (DOK-2)
- Modify functions to incorporate parameter(s). (DOK-2)
- Design App Lab applications that solve problems. (DOK-4)
- Apply the concept of good program design and good programming style in applications. (DOK-3)

Core Activities and Corresponding Instructional Methods:

- Write programs that utilize functions (including arguments and parameters), such as rectangular area, sorting two numbers, adding coins, Nim game, etc.
    - Lead a classroom discussion that prompts students to create accurate functions, arguments, parameters, and function returns.
    - Teacher led demonstration on how to incorporate functions to complete given tasks.
    - Direct instruction and practice on functions that incorporate arguments, parameters, and function returns.

Assessments:

- **Diagnostic:** Analysis of student work (e.g., classroom assignments, work samples, quizzes)/Observation and anecdotal notes
- **Formative:**
  - Diagnostic assessment and questioning
  - Unit 4 Quiz
  - Practice programming assignments
- **Summative:**
  - Graded programming assignments
  - Unit 4 Quiz

Unit 5: String Functions 10 days

Standards:

- *Pennsylvania Department of Education Computer Science Standards*
    - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
    - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
    - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
    - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
    - 3B.AP.14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
    - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
    - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
    - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
    - 1B-AP-08: Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
    - 1B-AP-11: Decompose (break down) problems into smaller, manageable sub problems to facilitate the program development process.
    - 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
    - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
    - 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
    - 3B-AP-14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3A-AP-15: Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.

    - 3A-AP-16: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instruction.
    - 3A-AP-17: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
    - 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

Objectives:

- Recognize String functions. (DOK-1)
- Evaluate expressions that manipulate strings. (DOK-2).
- Construct algorithms that utilize String functions. (DOK-3).
- Design App Lab applications that solve problems. (DOK-4)
- Apply the concept of good program design and good programming style in applications. (DOK-3)

Core Activities and Corresponding Instructional Methods:

- Write String function programs that incorporate String functions (including toUpperCase(), toLowerCase(), length, indexOf, substring(f), substring(f,l), replace, includes, charAt, etc.).
    - Direct instruction and practice on String functions and their proper use.
    - Teacher led demonstration on incorporating String functions into applications.
    - Guided practice: Include step-by-step written explanation of how to use each String function.
    - Classroom discussion and guided practice on proper String functions usage and implementation in coding.

  Programs include:
    - Traversing a word that checks if every letter is a consonant or a vowel
    - Finding out how many letters/words are in a sentence
    - Determining the first, middle, and last letter in a given word/phrase
    - Identifying the two/three letter initials of a given name
    - Printing a given name backwards

Assessments:

- **Diagnostic:** Analysis of student work (e.g., classroom assignments, work samples, quizzes)/Observation and anecdotal notes
- **Formative:**
  - Diagnostic assessment and questioning
  - Unit 5 Quiz
  - Practice programming assignments
- **Summative:**
  - Graded programming assignments
  - Unit 5 Quiz

Unit 6: Final Application and Poster 10 days

Standards:

- *Pennsylvania Department of Education Computer Science Standards*
    - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
    - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
    - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
    - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
    - 3B.AP.14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
    - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
    - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
    - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
    - 1B-AP-08: Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
    - 1B-AP-11: Decompose (break down) problems into smaller, manageable sub problems to facilitate the program development process.
    - 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
    - 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
    - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
    - 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
    - 3B-AP-14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.

    - 3A-AP-15: Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.
    - 3A-AP-16: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instruction.
    - 3A-AP-17: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
    - 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

<u>Objectives:</u>
- Design and create a program that utilizes key concepts from Units 1-5. (DOK-4)
- Apply the concept of good program design and good programming style in applications. (DOK-3)

<u>Core Activities and Corresponding Instructional Methods:</u>
- Write a final program (of student design) that utilizes key concepts learned from Units 1-5. The program should incorporate at least two decision structures, at least one control structure, two student written functions, and at least two String functions. The program cannot be one that was written in class or extra credit. Students can learn new concepts on their own, but detailed comments must be present for new content.
- Create a poster that directly states the title, goal(s) and purpose of final program, as well as displays both the code and application screen(s). In addition, a QR code should be displayed on the poster, allowing users to directly access your published project link when scanned.

<u>Assessments:</u>
- **Diagnostic:** Analysis of student work (e.g., classroom assignments, work samples, quizzes)/Observation and anecdotal notes
- **Formative:**
    - Diagnostic assessment and questioning
    - Units 1-5 Accumulative Quiz
    - Practice programming assignments
- **Summative:**
    - Units 1-5 Accumulative Quiz
    - Final Program
    - Final Poster

Primary Textbook(s) Used for this Course of Instruction

Name of Textbook: Speaking JavaScript: An In-depth Guide for Programmers

Textbook ISBN #: 978-1449365035

Textbook Publisher & Year of Publication: O'Reilly Media, 2015

Curriculum Textbook is utilized in (title of course): Computer Programming 1

Name of Textbook

- Rauschmayer, A. (2015). Speaking JavaScript: An in-depth guide for programmers. Sebastopol, CA: O'Reilly Media.
    - Free Online Text Available at http://speakingjs.com/es5/index.html

Checklist to Complete and Submit:

(Scan and email)

_____ Copy of the curriculum using the template entitled "Planned Instruction," available on the district website.

_____ The primary textbook form(s).

_____ The appropriate payment form, in compliance with the maximum curriculum writing hours noted on the first page of this document.

Each principal and/or department chair has a schedule of First and Second Readers/Reviewers. Each Reader/Reviewer must sign & date below.

First Reader/Reviewer Printed Name Christine Marcial

First Reader/Reviewer Signature: Christine Marcial Date: April 7, 2021

Second Reader/Reviewer Printed Name__________________________

Second Reader/Reviewer Signature _____________________________ Date__________