Perris Union High School District
Course of Study

A. COURSE INFORMATION

Course Title:
Computer Science Discoveries
- ❑ New
- ✓ Revised

Transcript Title/Abbreviation:
Previous: Exploring Computer Science
(To be assigned by Educational Services)

Transcript Course Code/Number:
108521/105522
(To be assigned by Educational Services)

Subject Area:
- ❑ Social Science
- ❑ English
- ❑ Mathematics
- ❑ Laboratory Science
- ❑ World Languages
- ❑ Visual or Performing Arts
- ✓ College Prep Elective
- ❑ Other

Is this classified as a Career Technical Education course?
- ❑ Yes
- ✓ No

Grade Level
- ❑ MS
- ❑ HS
- ❑ 5
- ❑ 6
- ❑ 7
- ✓ 8
- ✓ 9
- ✓ 10
- ✓ 11
- ✓ 12

Required for Graduation:
- ❑ Yes
- ✓ No

Meets UC/CSU Requirements?
- ✓ Yes
- ❑ No

Was this course *previously approved by UC* for PUHSD?
- ✓ Yes
- ❑ No

(Will be verified by Ed Services)

Credential Required to teach this course:
Math; Business; Industrial Technology Ed; Computer Concepts & Applica
To be completed by Human Resources only.
Introductory Computer Science
4/26/17
Signature **Date**

Meets "Honors" Requirements?
- ❑ Yes
- ✓ No

Meets "AP" Requirements?
- ❑ Yes
- ✓ No

Unit Value/Length of Course:
- ❑ 0.5 (half year or semester equivalent)
- ✓ 1.0 (one year equivalent)
- ❑ 2.0 (two year equivalent)
- ❑ Other:

Originally Submitted by: Timothy Poseley
Site: Heritage High School
Date: 1/12/2016

Approvals	Name/Signature	Date
Dir. Curriculum/Instruction		4/18/17
Asst. Sup. Educational Services		5.11.17
PUHSD Board		

Prerequisite(s) (REQUIRED):

None

Corequisite(s) (REQUIRED):

Algebra 1

Brief Course Description (REQUIRED):

Computer Science Discoveries is a code.org course being offered for the 2017-2018 school year. Students work in teams to develop computational thinking and problem solving skills. The course does not aim to teach mastery of a single programming language but aims instead to develop computational thinking, to generate excitement about the field of computing, and to introduce computational tools that foster creativity. The course also aims to build students' awareness of the tremendous demand for computer specialists and for professionals in all fields who have computational skills. Each unit focuses on one or more computationally intensive career paths. The course also aims to engage students to consider issues raised by the present and future societal impact of computing.

B. COURSE CONTENT

Course Purpose (REQUIRED):
What is the purpose of this course? Please provide a brief description of the goals and expected outcomes. Note: More specificity than a simple recitation of the State Standards is needed.

Computer Science Discoveries is designed to introduce students to the breadth of the field of computer science through an exploration of engaging and accessible topics. Rather than focusing the entire course on learning particular software tools or programming languages, the course is designed to focus the conceptual ideas of computing and help students understand why certain tools or languages might be utilized to solve particular problems. The goal of Computer Science Discoveries is to develop in students the computational thinking practices of algorithm development, problem solving and programming within the context of problems that are relevant to the lives of today's students. Students will also be introduced to topics such as interface design, limits of computers and societal and ethical issues.

Course Outline (REQUIRED):
Detailed description of topics covered. All historical knowledge is expected to be empirically based, give examples. Show examples of how the text is incorporated into the topics covered.

Unit 1 - Problem Solving: Computers and Logic (3 weeks)

Topics to be addressed:
Lesson 1: Intro to Problem Solving
Lesson 2: The Problem Solving Process
Lesson 3: Exploring Problem Solving
Lesson 4: What is a Computer?
Lesson 5: Representing Information
Lesson 6: Processing with Bits
Lesson 7: Human vs. Computer Processing
Lesson 8: Processing with Apps
Lesson 9: Apps and Problem Solving

Topic Description:
Computers are all around us, and for students much of their everyday action is impacted by computing. In the unit students will explore what it means for something to be a computer - what core functionality brings together all these items we think of as computers. Students should know how to use computers effectively - this means being able to navigate a computer and accomplish tasks. Students look at the many things computers allow people to do.

Objectives:
The students will be able to:

- Communicate and collaborate with classmates in order to solve a problem
- Identify the four steps of the problem solving process
- Apply the problem solving process to approach a variety of problems
- Identify a computer as a machine that processes information
- Create and use a representation system to store and retrieve information
- Use a bit level representation system to process information.
- Develop, articulate, and implement a method for processing information based on given constraints
- Identify the information that an app uses as input or produces as output
- Identify and define a problem that could be solved using computing

Unit 2: Problem Solving (4 weeks)
Topics to be addressed:
Lesson 1: Quality Websites
Lesson 2: Website Design
Lesson 3: Describing Web Pages
Lesson 4: Text on the Web
Lesson 5: Images in HTML
Lesson 6: Clean Code and Debugging
Lesson 7: Styling Text Elements with CSS
Lesson 8: Styling Elements with CSS
Lesson 9: Multi-Page Websites
Lesson 10: RGB Colors and Classes
Lesson 11: Peer Review: Personal Portfolio Website
Lesson 12: Digital Footprint
Lesson 13: Website Review and Publish

Lesson 14: Sources and Search Engines
Lesson 15: Intellectual Property
Lesson 16: Project: Personal Portfolio Website
Lesson 17: Sending Information Between Nodes
Lesson 18: Addressing
Lesson 19: Sending Larger Amounts of Information

Topic Description:
Students will find empowerment through the realization that they too can take part in this creation and dissemination of information online by developing their own web pages from scratch using just HTML and CSS. By critically examining the web pages they visit every day, students will start to explore what makes a strong website and then use their new design process to implement parts of those designs.

Objectives:
The students will be able to:

- Identify the purpose of a website and explain how well it is meeting that purpose
- Create their own language that describes the structure and content of a web page using just letters, basic symbols and numbers
- Describe what HTML is and why it's important to have a common language for creating web pages
- Use CSS selectors to style HTML text elements
- Connect multiple web pages into one website using hyperlinks
- Give constructive feedback to peers on their websites
- Understand and explain reasons that it is difficult to control who sees information published online.
- Identify the impacts of sharing specific personal information online and consider what level of sharing is appropriate
- Explain how search engines look for relevance and trustworthiness of website when searching
- Explain the purpose of copyright

Unit 3 - Programming: Interactive Games and Animations (5 weeks)
Topics to be addressed:
Lesson 1: Drawing in Game Lab
Lesson 2: Draw Loop and Randomization
Lesson 3: Variables Unplugged
Lesson 4: Variables and Animation
Lesson 5: Sprites and Properties
Lesson 6: Sprites and Images
Lesson 7: Booleans and Conditionals
Lesson 8: Conditionals and Keyboard Input
Lesson 9: Complex Conditionals
Lesson 10: Project: Interactive Card
Lesson 11: Velocity
Lesson 12: Collision Detection
Lesson 13: Complex Sprite Movement
Lesson 14: Collisions
Lesson 15: Functions

Lesson 16: The Game Design Process
Lesson 17: Platform Jumper
Lesson 18: Project - Design a Game

Topic Description:
Starting off with simple primitive shapes and building up to more sophisticated sprite-based games, students will become familiar with the basic concepts that form the foundation of computer programming. The development of a personalized final project will engage students in design, testing, and iteration as they come to see that failure and debugging are an expected (and valuable) part of the programming process that make your end product better.

Objectives:
The students will be able to:

- Use clean coding practices by grouping code into at least 3 logical chunks with comments
- Write a short program in pseudocode that manipulates values in memory
- Use variables in a program to store information that is used multiple times
- Assign images to sprites
- Use nested conditionals to make decision based off of two conditions.
- Use the velocity and rotation Speed blocks to create and change sprite movements
- Create and use functions for blocks of code that perform a single high-level task within a program
- Identify core programming constructs necessary to build different components of a game
- Independently scope the features of a piece of software

Unit 4 - Problem Solving: The Design Process (6 weeks)
Topics to be addressed:
Lesson 1: Analysis of Design
Lesson 2: Understanding Your User
Lesson 3: User-Centered Design Micro Activity
Lesson 4: User Interface and Prototype Testing
Lesson 5: Feedback and Prototypes
Lesson 6: Identifying User Needs
Lesson 7: Project - Paper Prototype
Lesson 8: Designing Apps for Good
Lesson 9: Market Research
Lesson 10: Paper Prototypes
Lesson 11: Prototype Testing
Lesson 12: Digital Design
Lesson 13: Event Driven Programming
Lesson 14: Basic App Functionality
Lesson 15: Testing the App
Lesson 16: Improving and Iterating
Lesson 17: App Presentation

Topic Description:
With the Problem Solving Process as a basis, students will start to look at how user-centered design places _others_ first in approaching the solution to a problem, whether through design of a physical object or a software application.

Through several small projects, students will experience an iterative design and testing process that will then be applied to a multi-week group project. In this final project, student teams will identify an audience, investigate their needs, and develop a concept and paper wireframe for an app to serve that audience. With concept and wireframe in hand, teams will then develop an interactive prototype of their apps that they can use to test with actual users, taking feedback to drive further development.

Objectives:
Students will be able to:

- Express opinions respectfully and effectively
- Distinguish between their own needs and the needs of their users
- Empathize with a user's needs to design an object
- Translate user needs into changes and improvements in the user interface of an app
- Design the functionality of an app to address the specific needs of a user
- Identify ways in which apps can effect social change
- Document user interactions with a prototype of an app's User Interface using paper and pen
- Write programs that respond to user input
- Apply the event driven programming model to new events
- Analyze the user feedback from the previous lesson and determine a list of bugs (flaws) that need to be fixed and features that could be added to the app
- Present technical information clearly to non-technical users

Unit 5: The Internet - Data and Society (6 weeks)
Topics to be addressed:
Lesson 1: Representation Matters
Lesson 2: Patterns and Representation
Lesson 3: ASCII and Binary Representation
Lesson 4: Representing Images
Lesson 5: Binary Numbers Part 1
Lesson 6: Binary Numbers Part 2
Lesson 7: Combining Representations
Lesson 8: Create Representation
Lesson 9: Problem Solving and Data
Lesson 10: Making Decisions with Data
Lesson 11: Interpreting Data
Lesson 12: Automating Data Decisions
Lesson 13: Problem Solving with Big Data
Lesson 14: Project - Solve a Data Problem

Topic Description:
Exploring the importance of data in solving problems and highlighting the role of computers in this process, students learn systems commonly used to represent text, images, and numbers on a computer and explore the commonalities between them. Students look at how the problem solving process is applied in the context of data. Students learn that data needs to be collected, cleaned, and interpreted before it can be used for making decisions. This unit highlights how both computers and humans approach the different steps in this process. Concludes by considering how the data problem solving process could be applied to an area of the student's choosing.

Objectives:
The students will be able to:

- Define data as information collected from the world to help make a recommendation or solve a problem
- Describe the necessary features of a system for representing information
- Define a binary system as one that uses just two possible states to represent information
- Use a binary system to represent numbers
- Determine the most appropriate encoding system for a given piece of information.
- Choose and justify the use of different binary representation systems depending on the information being represented
- Use the problem solving process to answer a question using data
- Identify and remove irrelevant data from a data set
- Design an algorithm for making decisions using data as inputs
- Give examples of how data is collected from sensors and tracking user behavior
- Apply the data problem solving process to a personally relevant topic

Unit 6 - Programming: The Internet of Things (7 weeks)

Topics to be addressed:

Lesson 1: Innovations in Computing
Lesson 2: Input Unplugged
Lesson 3: Event Types
Lesson 4: Getters and Setters
Lesson 5: The Circuit Playground
Lesson 6: Lists
Lesson 7: Color LEDs
Lesson 8: For Loops
Lesson 9: Lists and For Loops
Lesson 10: Timed Loops
Lesson 11: Project: Board Output
Lesson 12: Physical Input
Lesson 13: Analog Input
Lesson 14: Sensor Applications
Lesson 15: Project: Prototype an Innovation

Topic Description:

Computing devices are everywhere, and we often engage with them without even realizing it. The ubiquitous availability of the internet and continual shrinking of computers have opened up new opportunities for innovations in developing classes of computing including smart appliances, wearable technology, biohacking, and rapid prototyping.

By exploring innovative computing devices from a variety of fields, students will explore the essential elements of computer hardware. Using a bare microcontro board with several integrated sensors and output devices students will learn how software interacts with hardware and they'll develop prototypes of physical computing devices.

In the final stage of the course student groups will once again return to their capstone apps, this time connecting them with their physical computing boards as a means of input, output, or both.

Objectives:
The students will be able to:

- Identify computing innovations within a given field
- Compare and contrast multiple ways to take input
- Create a user interface composed of multiple design elements
- Use a getter to get the current content of a UI element
- Connect and troubleshoot external devices
- Understand how to use a for loop as a way to repeat a set of code a certain number of times
- Use event handlers to take user input
- Attach an event handler to a hardware input
- Develop programs that respond to analog input
- Develop apps that take input through analog sensors
- Independently scope the features of a piece of software

Writing Assignments (REQUIRED):
Give examples of the writing assignments and the use of critical analysis within the writing assignments.

Writing assignments are provided by code.org once teachers are trained.

INSTRUCTIONAL MATERIALS (REQUIRED)	
Textbook #1	
Title:	Edition:
Author:	ISBN:
Publisher:	Publication Date:
Usage: ❑ Primary Text ❑ Read in entirety or near	
Textbook #2	
Title:	Edition:
Author:	ISBN:
Publisher:	Publication Date:

Usage: ❑ Primary Text ❑ Read in entirety or near	

Supplemental Instructional Materials *Please include online, and open source resources if any.*
Provided resources are all located in code.org

Estimated costs for classroom materials and supplies (REQUIRED). *Please describe in detail.* If more space is needed than what is provided, please attach backup as applicable.	
Cost for class set of textbooks: $	Description of Additional Costs:
Additional costs:$	
Total cost per class set of instructional materials:	$

Key Assignments (REQUIRED):
Please provide a detailed description of the Key Assignments including tests, and quizzes, which should incorporate not only short answers but essay questions also. How do assignments incorporate topics? Include all major assessments that students will be required to complete
Since the class is primarily hands-on, project-based learning experience for the students, there will be very little traditional learning methodologies such as worksheets, problems from the book assigned as homework, etc. Most learning will happen during the multi-day projects themselves, although students will be taught what overall learning goals for each unit, as well as specific methodologies for completing the tasks using the appropriate technology. Unit 1: 1. Processing with Apps a. This lesson covers the input and output aspects of computers in a context that is relevant and familiar to students: apps. In pairs, students will evaluate various web applications to analyze the specific problems that they were designed to solve, the inputs that they need to work, as well as the outputs they provide to users. The class will conclude with observations of these apps as well as a teacher led discussion about the impact of apps on society. 2. Apps and Problem Solving a. To conclude their study of the problem solving process and the input/output/store/process model of a computer, students will propose an app designed to solve a real world problem. This project will be completed across multiple days and will result in students creating a poster highlighting the features of their app that they will present to their classmates. A project guide provides step by step instructions for students and helps them organize their thoughts. The project is designed to be completed in pairs though it can be completed individually. Unit 2: This unit will incorporate many lessons and projects, mostly related to website design and building. The students will cover general structure of a website, using the peer review process for a portfolio website, and then go through a

rubric scoring process. Students will also share their final website and the process used to create it.

1. Clean Code and Debugging
    a. Students will learn about common issues that arise when designing web pages in HTML. Students will correct errors in a sequence of increasingly complex web pages found on Code Studio. In the process they will learn the importance of comments, whitespace, and indentation as tools for making web pages easier to read. At the end of the lesson students create a list of strategies for debugging web pages and ensuring they are easy to read and maintain.
2. Peer Review: Personal Portfolio Website
    a. Students have been continuously working on the same project for many weeks. In this lesson students take a step back from implementing their own website and look at peers websites. Through peer review students will get to share feedback on each others sites as well as hopefully get some new ideas for their own website.
3. Project: Personal Portfolio Website
    a. Students have spent a lot of time throughout the unit working on their Personal Website. In the final couple of days students finalize their websites. They work with peers to get feedback, put the finishing touches on the websites, review the rubric and reflect on their process. To cap off the unit, they will share their projects and also a overview of the process they took to get to that final design.

Unit 3:

1. The Game Design Process
    a. This lesson introduces students to the process they will use to design games. This process is centered around a project guide which asks students to define their sprites, variables, and functions before they begin programming their game. In this lesson students begin by playing a game on Game Lab where the code is hidden. They discuss what they think the sprites, variables, and functions would need to be to make the game. They are then given a completed project guide which shows one way to implement the game. Students are then walked through this process through a series of levels. As part of this lesson students also briefly learn to use multi-frame animations in Game Lab. At the end of the lesson students have an opportunity to make improvements to the game to make it their own.
2. Using the Game Design Process
    a. Students will use the problem solving process from Unit 1 to create a platform jumper game. They will start by looking at an example of a platform jumper, then define what their games will look like. Next, they will use a structured process to plan the backgrounds, variables, sprites, and functions they will need to implement their game. After writing the code for the game, students will reflect on how the game could be improved, and implement those changes.
3. Design a Game
    a. Students will plan and build their own game using the project guide from the previous two lessons to guide their project. Working individually or in pairs, students will first decide on the type of game they'd like to build, taking as inspiration a set of sample games. They will then complete a blank project guide where they will describe the game's behavior and scope out the variables, sprites, and functions they'll need to build. In Code Studio, a series of levels prompts them on a general sequence they can use to implement this plan. Partway through the process, students will share their projects for peer review and will incorporate feedback as they finish their game. At the end of the lesson, students will share their completed games with their classmates.

Unit 4:

1. Testing the App
    a. By the end of the last lesson the students should have a minimum viable prototype of their app. The primary purpose of this lesson is to have the team actually test the app with other people, preferably from the target audience the app is intended for, or from different teams in the class while observers from the team will record the results on the worksheets they used in the planning phase. As with testing the paper prototypes, teams will start by planning for the specific scenarios before running

and observing tests.

2. Improving and Iterating
    a. Teams have at this point developed an app prototype that has gone through multiple iterations and rounds of user testing. With the information and guidance gained from the last round of user testing, each student will have the opportunity to plan for and implement improvements to the team app.
3. App Presentation
    a. At this point teams have researched a topic of personal and social importance, developed and tested both a paper prototype and a digital prototype, and iterated on the initial app to incorporate new features and bug fixes. Now is the time for them to review what they have done and pull together a coherent presentation to demonstrate their process of creation.

Unit 5:

1. Automating Data Decisions
    a. In this lesson students look at a simple example of how a computer could be used to complete essentially every step of the data problem solving process. To begin students are given the clean data they collected about their ice cream preferences. They use a spreadsheet to visualize this data using simple charts and discuss in groups the decision they would make with this data. Students then create rules, or an algorithm, that a computer could use to make this decision automatically. Groups share their rules and what choices their rules would make with the class data. They then use their rules on data from another class to reinforce the fact that the rules they create should be useful for any data set. The lesson concludes with a discussion about the benefits and drawbacks of using computers to automate the data problem solving process.
2. Problem Solving with Big Data
    a. In this lesson, students look at how data is collected and used by a organizations to solve problems in the real world. The lesson begins with a quick review of the data problem solving process they have explored in this unit. Then students are presented three scenarios that could be solved using data and they brainstorm the types of data they would want to solve them and how they could collect the data. Each problem is designed to reflect a real-world service that exists. After brainstorming students will watch a video about a real-world service and will record notes about what data is collected by the real-world service and how it is used. At the end of the lesson students record whether data was provided actively by a user, was recorded passively, or is collected by sensors.
3. Project - Solve a Data Problem
    a. To conclude this unit students will design a way to use data to make a recommendation or prediction to help solve a problem. Students will follow a project guide to complete this multi-day activity. In the first several steps students brainstorm problems, perform simple research, and define a problem of their choosing. They will then decide what kind of data they would want to collect, how it could be collected, and how it could be used. Students will perform a peer review and make any necessary updates to their project before preparing a presentation to their classmates.

Unit 6:

1. Analog Input
    a. Students explore the three analog sensors on the Circuit Playground which measure sound, light, and temperature. This lesson introduces analog sensors as a new form of input. These sensors all perform analog-to-digital conversion, allowing programs to sense things as represented by a 10 bit number. Students will be able to develop programs that respond to analog input, scale a range of numbers to meet a specific need, and represent a sensor value in a variety of ways.
2. Sensor Applications
    a. Students develop three small apps, each of which will rely on one analog sensor for key functionality. For each of the apps, students will identify all of the inputs and outputs that the program will need, include both UI elements and board elements. Students will describe how the inputs and outputs will interact.

3. Project - Prototype an Innovation
    a. In this final project for the course, student team to develop and test a prototype for an innovative computing device based on the Circuit Playground. Using the inputs and outputs available on the board, groups will create programs that allow for interesting and unique user interactions.

Instructional Methods and/or Strategies (REQUIRED):

Please list specific instructional methods that will be use.

There are several concrete instructional strategies that are included in each unit to implement this culturally relevant, inquiry-based vision.

- Each unit begins with a description of the topic, an explanation of the importance of this topic, possible social applications of this topic, and objectives for the unit.
    - CCSS:
    - Mathematical Practice:
        - Construct viable arguments and critique the reasoning of others.

- Units typically begin with a kinesthetic activity to get students involved in the unit topic. Students are more engaged when they go beyond seatwork to gain familiarity with the scope of a topic. Acting out computing concepts is one way to have students actively engaged in the curriculum.
    - CCSS:
    - Mathematical Practice:
        - Make sense of problems and persevere in solving them.

- In most units, the final unit project is presented at the beginning of the unit so students understand what type of project they will engage in at the end of the unit. Daily assignments help scaffold their knowledge towards gaining the knowledge needed to complete a particular project. The final project represents a culmination of their new knowledge and provides an opportunity to expand their understandings to a particular socially-relevant problem.
    - CCSS:
    - Mathematical Practice:
        - Make sense of problems and persevere in solving them.
        - Use appropriate tools strategically.

- Computing terms and definitions are explicit and part of the instruction. The curriculum avoids unnecessary jargon, which might distract from learning of the critical content. Students have opportunities to use writing to reinforce the literacy component behind these computing terms and definitions.
    - CCSS:
    - Mathematical Practice:
        - Construct viable arguments and critique the reasoning of others.

- Foundational computing topics are connected to the 'pop-technology' students have likely encountered: mobile phones, social networks, blogs, Internet browsing, etc.
    - CCSS:
    - Mathematical Practice:
        - Construct viable arguments and critique the reasoning of others.

- Real world problems are presented in the context of socially-relevant issues impacting urban communities (housing, safety, poverty, health care, access to equal rights, educational opportunities, improving social

services, translation services, transportation, etc.)
    - CCSS:
    - Mathematical Practice:
        - Construct viable arguments and critique the reasoning of others.
        - Make sense of problems and persevere in solving them.

- Students have opportunities to work on problems that they help define and can individualize—i.e. selecting their own content for websites; creating original, not pre-scripted, problem-solving strategies, etc.
    - CCSS:
    - Mathematical Practice:
        - Construct viable arguments and critique the reasoning of others.
        - Make sense of problems and persevere in solving them.
        - Use appropriate tools strategically.

- Activities are designed to encourage students to work in a variety of collaborative settings including elbow partners, peer-programming, and group research projects. This collaboration encourages conversations around computing topics.
    - CCSS:
    - Mathematical Practice:
        - Construct viable arguments and critique the reasoning of others.
        - Make sense of problems and persevere in solving them.
        - Use appropriate tools strategically.
        -
        -
- Students will experience a variety of ways to communicate their answers—academic writing, journal entries, writing a letter to a friend or companion, using presentation software, developing graphics or animation, storyboarding, listing algorithms, drawing illustrations, oral presentations, etc.
    - CCSS:
    - Mathematical Practice:
        - Construct viable arguments and critique the reasoning of others.

- Units incorporate examples of careers in computing as they arise in the curriculum. Students will be given hypothetical opportunities to act as a professional to take on the behavior and skills to solve a given problem.
- Although using technology is a core component of this curriculum, using computers is not necessarily embedded in the curriculum on a daily basis.

- All of these strategies contribute to developing the problem-solving skills and computational practices that are emphasized throughout the course.

- It is important to note that each unit focuses on different instructional strategies; this is purposeful. In some cases, it is because the particular subject matter lends itself more successfully to a particular set of strategies, but this was also done to highlight the wide variety of possible strategies that can be used effectively in teaching this course. We encourage teachers to experiment by trying strategies that work well for them in a variety of different places in the curriculum. Journal responses and blog entries can be used by students to communicate about their work in any of the units. Peer reviews, gallery walks, jigsaws, role-plays and collaborative groups of varying sizes can be used for activities throughout the course.
    - CCSS:

- Mathematical Practice:

Construct viable arguments and critique the reasoning of others.

Assessment Methods and/or Tools (REQUIRED):

Please list different methods of assessments that will be used.

Each unit will incorporate multiple assessment strategies. Since the class is primarily hands-on, project-based education, the majority of assessments will be project-based. Generally each unit will have several projects. Some units will have several discrete projects, each one reflecting on a certain portion of the unit. Other units will have a scaffolded series of projects, culminating in a major project that demonstrates the students accumulated knowledge and abilities in each unit. Formative assessments may include quizzes, small projects early in a unit, and day-to-day student-teacher interaction. Final projects in each unit will be scored as their summative assessments; project-based education models prefer a hands-on approach to demonstrate student achievement. Successful completion of a project and achievement of coals indicate proficiency levels of a particular unit's goals.

Unit 1:

This unit primarily aligns with CCSS Standards:

Mathematical Practice:

- Make sense of problems and persevere in solving them.
- Construct viable arguments and critique the reasoning of others.
- Model with Mathematics.
- Use appropriate tools strategically.

Formative assessments will include scoring the "buy a computer" project. This project will be graded on each student's knowledge of computer hardware, software, and the ability to determine needs and match a system with the requirements for that system, as well as their ability to perform specific internet searches and evaluate the results of those searches.

- CCSS: Writing Standards for Literacy in History/Social Studies, Science, and Technical Subjects
  - 6–12 - Grades 9-10 students: 8. Gather relevant information from multiple authoritative print and digital sources, using advanced searches effectively; assess the usefulness of each source in answering the research question; integrate information into the text selectively to maintain the flow of ideas, avoiding plagiarism and following a standard format for citation.

Another formative assessment will be the "communications data" project. This project will be graded on each students ability to collect meaningful data, analyze and interpret data with respect to intelligent behavior patterns, and communicate their understanding of computer-based communications as data exchange.

- CCSS: Mathematical Practice:
  - Construct viable arguments and critique the reasoning of others.

For a Unit 1 summative assessment students will be graded on their post-test analysis of their having a computer take the Turing Test. The students' performance on the test is immaterial as no one can "pass" or "fail" the test; it is their analysis of the results and how the computer performed that will be assessed. This will check their understanding of data, models of intelligent behavior, computer communications, and the societal impact of computing and data exchange.

- CCSS: Mathematical Practice:
- Make sense of problems and persevere in solving them.

- Model with mathematics.
- Use appropriate tools strategically.
- Construct viable arguments and critique the reasoning of others.

Unit 2:
This unit primarily aligns with CCSS Standards:
Mathematical 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.

Mathematical Content
- Building Functions F-BF 1a - Build a function that models a relationship between two quantities - Write a function that describes a relationship between two quantities: Determine an explicit expression, a recursive process, or steps for calculation from a context.
- Building Functions F-BF 1a - Write a function that describes a relationship between two quantities: Determine an explicit expression, a recursive process, or steps for calculation from a context.

Formative assessments will include scoring the fencepost problem, a follow-on to the handshake activity. This will measure students' ability to deconstruct and analyze a problem and create an algorithmic solution for (n) instances.
- CCSS: Mathematical Practice:
- Make sense of problems and persevere in solving them.
- Construct viable arguments and critique the reasoning of others.
- Reason abstractly and quantitatively.
- Model with Mathematics Mathematical Content:

• Building Functions F-BF: 1a Build a function that models a relationship between two quantities - Write a function that describes a relationship between two quantities: Determine an explicit expression, a recursive process, or steps for calculation from a context.
Another formative assessment to be scored will be the Tower project, a follow-on to the Dots activity. These activities introduce students to binary numbers and simple binary counting algorithms, an important next step in creating binary search and sorting algorithms.
- CCSS: Mathematical Practice:
- Make sense of problems and persevere in solving them.
- Construct viable arguments and critique the reasoning of others.
- Reason abstractly and quantitatively.

Model with Mathematics Mathematical Content:
- Building Functions F-BF: 1a Build a function that models a relationship between two quantities - Write a function that describes a relationship between two quantities: Determine an explicit expression, a recursive process, or steps for calculation from a context.

For a Unit 2 summative assessment students will create a shortest route map. Students will build on their knowledge and skills in creating algorithms and create a shortest route (minimum spanning tree) algorithm

based on personal data collected. This will test their ability to create multiple algorithms from a dataset of unknown (n) values and determine the most efficient solution.

- CCSS: Mathematical Practice:
- Make sense of problems and persevere in solving them.
- Construct viable arguments and critique the reasoning of others.
- Reason abstractly and quantitatively.

Model with Mathematics Mathematical Content:

- Building Functions F-BF: 1a Build a function that models a relationship between two quantities - Write a function that describes a relationship between two quantities: Determine an explicit expression, a recursive process, or steps for calculation from a context.

Unit 3:

This unit primarily aligns with CCSS Standards:
Mathematical Practice:

- Make sense of problems and persevere in solving them.
- Construct viable arguments and critique the reasoning of others.

Formative assessments for basic html will include the creation of a simple html web page with certain specific requirements; another will be a quiz on basic html tags and structures. Student analyses of existing web sites will be assessed to determine students understanding of web site objectives and purposes.

- CCSS: Mathematical Practice:
- Construct viable arguments and critique the reasoning of others.

Another formative assessment will be to create or select the css styles to produce certain predefined output requirements.

For a Unit 3 summative assessment students will produce a complete web site to support a fictitious business. Students will create a template, lay our navigation elements, select graphics, produce a css style sheet, create content and html pages. This will assess students ability to create web pages which are stylistically and thematically appropriate, incorporate all relevant design strategies and Dreamweaver techniques, and connect the site and its content to a set of relevant requirements.

- CCSS: Mathematical Practice:
- Make sense of problems and persevere in solving them.
- Construct viable arguments and critique the reasoning of others.

Unit 4:

This unit primarily aligns with CCSS Standards:
Mathematical Practice:

- Use appropriate tools strategically.
- Make sense of problems and persevere in solving them.

Mathematical Content

- Building Functions F-BF 1b Build a function that models a relationship between two quantities - Write a function that describes a relationship between two quantities: Combine standard function types using arithmetic operations.

· Creating Equations A-CED 3 - Create Equations that describe numbers or relationships: Represent constraints by equations or inequalities, and by systems of equations and/or inequalities, and interpret solutions as viable or nonviable options in a modeling context.

Formative assessments include scoring a simple program with Scratch that manipulates a single sprite, followed by a more sophisticated program controlling multiple sprites. CCSS: • Mathematical Practice: Use appropriate tools strategically.
These will test students' ability to understand and implement simple repetitive algorithmic routines. Another scored assessment will be the broadcast event project. This will test students' ability to write event-driven code and sprite responses.
· CCSS: Mathematical Practice:
• Make sense of problems and persevere in solving them.
Mathematical Content:
• Building Functions F-BF 1b Build a function that models a relationship between two quantities - Write a function that describes a relationship between two quantities: Combine standard function types using arithmetic operations.
This unit will likely have at least one quiz testing students' knowledge of facts about Scratch and analysis of existent Scratch programs. Another scored assessment will be the RPS project, which will test students; knowledge of variables and conditionals.
· CCSS: Mathematical Practice:
• Make sense of problems and persevere in solving them.
Mathematical Content:
• Creating Equations A-CED 3 - Create Equations that describe numbers or relationships: Represent constraints by equations or inequalities, and by systems of equations and/or inequalities, and interpret solutions as viable or nonviable options in a modeling context.

For the Unit 4 summative assessment students will develop a complete Scratch game. This will incorporate all students' knowledge and skills incorporated so far, including looping, conditionals, multi-sprite interactions, multiple scenes, etc. Students will be scored on adherence to requirements, appropriateness of content, game completeness, and program correctness (i.e. absence of bugs).
· CCSS: Mathematical Practice:
• Make sense of problems and persevere in solving them.
Mathematical Content:
• Creating Equations A-CED 3 - Create Equations that describe numbers or relationships: Represent constraints by equations or inequalities, and by systems of equations and/or inequalities, and interpret solutions as viable or nonviable options in a modeling context.
• Using Mathematics and Computational Thinking - Use simple limit cases to test mathematical expressions, computer programs or algorithms or simulations to see if a model "makes sense" by comparing the outcomes with what is known about the real world.

Unit 5:
This unit primarily aligns with CCSS Standards:
Mathematical Practice:
· Construct viable arguments and critique the reasoning of others.

Mathematical Content

· Interpreting Categorical and Quantitative Data S-ID 1 - Summarize, represent, and interpret data on a single count or measurement variable: Represent data with plots on the real number line (dot plots, histograms, and box plots).

· Interpreting Categorical and Quantitative Data S-ID 3 - Summarize, represent, and interpret data on a single count or measurement variable: Interpret differences in shape, center, and spread in the context of data sets, accounting for possible effects of extreme data points (outliers).

· Conditional Probability and the Rules of Probability S-CP 1 - Understand independence and conditional probability and use them to interpret data: Describe events as subsets of a sample space (the set of outcomes) using characteristics (or categories) of the outcomes, or as unions, intersections, or complements of other events ("or" and "not").

· Conditional Probability and Rules of Probability S-CP 1 - Describe events as subsets of a sample space (the set of outcomes) using characteristics (or categories) of the outcomes, or as unions, intersections, or complements of other events ("or", "and", "not").

· Making inferences and Justifying Conclusions S- IC 4 - Make inferences and justify conclusions from sample surveys, experiments, and observational studies: Use data from a sample survey to estimate a population mean or proportion; develop a margin of error through the use of simulation models for random sampling.

Formative assessments include scoring the data collection project; particularly students' data collection vs. the requirements. An analysis of the data will be a separate assessment. A follow-up summative assessment will be a quiz presenting students with certain sets of data and having students analyze that data using a variety of different techniques.

· CCSS: Mathematical Practice:

• Construct viable arguments and critique the reasoning of others.

Mathematical Content:

· Interpreting Categorical and Quantitative Data S-ID 1 - Summarize, represent, and interpret data on a single count or measurement variable: Represent data with plots on the real number line (dot plots, histograms, and box plots).

· Conditional Probability and the Rules of Probability S-CP 1 - Understand independence and conditional probability and use them to interpret data: Describe events as subsets of a sample space (the set of outcomes) using characteristics (or categories) of the outcomes, or as unions, intersections, or complements of other events ("or" and "not").

A separate quiz over the appInventor software and its usage will also be included, testing students' ability to read and understand program snippets and analyze outcomes and outputs.

For the Unit 5 summative assessment students will develop an Android program using appInventor that collects and analyzes some pre-defined data. Students will be scored on the correctness of the program and its usability, the applicability of the student developed data analysis algorithm and its fulfillments of the predefined requirements, and the effectiveness of the presentation of the data analysis tool.

· CCSS: Mathematical Practice:

• Construct viable arguments and critique the reasoning of others.

Mathematical Content:

• Making inferences and Justifying Conclusions S-IC 4 - Make inferences and justify conclusions from sample surveys, experiments, and observational studies: Use data from a sample survey to estimate a population mean or proportion; develop a margin of error through the use of simulation models for random sampling.

Unit 6:
This unit primarily aligns with CCSS Standards:
Mathematical Practice:
- Make sense of problems and persevere in solving them
- Construct viable arguments and critique the reasoning of others.
- Use appropriate tools strategically

Mathematical Content
- Creating Equations A-CED 3 - Create Equations that describe numbers or relationships: Represent constraints by equations or inequalities, and by systems of equations and/or inequalities, and interpret solutions as viable or nonviable options in a modeling context.

Formative assessments include a short quiz testing students' knowledge of robotics basics and current applications. Another assessment will be a quiz testing students' knowledge of NXT software and common controls, as well as basic program structures. Another assessment will be the robotics tic-tac-toe game project, where students will be scored on ability to correctly program the tic-tac-toe algorithm in the LabView NXT programming environment.
- CCSS:

Mathematical Practice:
• Make sense of problems and persevere in solving them.
Mathematical Content:
• Creating Equations A-CED 3 - Create Equations that describe numbers or relationships: Represent constraints by equations or inequalities, and by systems of equations and/or inequalities, and interpret solutions as viable or nonviable options in a modeling context.

For the unit 6 summative assessment students will design, build, and program a dancing robot. Robots will perform choreographed dances to pre-recorded music. Scoring will be based on program completeness and correctness, choreography, robot construction, and quality of presentation.
- CCSS:

Mathematical Practice:
- Construct viable arguments and critique the reasoning of others.
- Make sense of problems and persevere in solving them.

COURSE PACING GUIDE AND OBJECTIVES (REQUIRED)				
Day(s)	**Objective**	**Standard(s)**	**Chapter(s)**	**Reference**
	Provided by RCOE once trained.			

C. HONORS COURSES ONLY
Indicate how much this honors course is different from the standard course.

D. BACKGROUND INFORMATION
Context for course (optional)
History of Course Development (optional)