El Monte Union High School District

Course Outline

High School Rosemead High School
January 2016

Title: Intro to Comp Sci_____

Transitional*_______(Eng. Dept. Only)

Sheltered (SDAIE)*___Bilingual*___

AP**_______Honors**__________

Department: Mathematics/IT_______

Grade Level (s):_____9-12________

Semester__________Year__X____

Year of State Framework Adoption _______

This course meets graduation requirements:

( ) English
( ) Fine Arts
( ) Foreign Language
( ) Health & Safety
( ) Math
( ) Physical Education
( ) Science
( ) Social Science
(x) Elective

Department/Cluster Approval	Date
_______________	__________
_______________	__________
_______________	__________
_______________	__________
_______________	__________
_______________	__________

*Instructional materials appropriate for English language learners are required.

**For AP/Honors course attach a page describing how this course is above and beyond a regular course. Also, explain why this course is the equivalent of a college level class.

1. Prerequisite(s):
   - 9th graders
   - C or Above grade in 8th grade Math class
   - Math SBA Proficiency level: Standards Nearly Met of Standards Not Met

2. Short description of course which may also be used in the registration manual:

Introduction to Computer Science (ICS) is designed to be implemented as a full-year high school course using a PLTW Gateway modules. In each module, student teams create an Android® interface to solve a problem the team defines. Students learn fundamental computer science (CS) concepts using MIT App Inventor.

The course aims to develop computational thinking and build student excitement. Several days in each module are targeted to build career awareness about computing skills in all fields and to improve students' cyber hygiene.

3. Describe how this course integrates the schools ESLRS (Expected Schoolwide Learning Results):

This course integrates Rosemead’s ESLRs by combining elements of hands on career technical education in various areas of material working with the fundamental skills of reading, writing and mathematics. Furthermore, concepts from the computer sciences will be used during on-going instruction to facilitate student observation and learning of fundamental common practices in computer software design, computer engineering problem solving and software production processes. Using computer-based design technology will be used along with cutting programming technology. Assessment methods will be based on project evaluations specifically involving a measurement of the degree to which students follow written instructions in the execution of the operations involved in coding and documentation of projects Performance or activity-based assessments will also be utilized to grade students.

4. Describe the additional efforts/teaching techniques/methodology to be used to meet the needs of English language learners:

Additional efforts, techniques and methodologies used to meet the needs of ELL students include the pairing or partnering of ELL students with bilingual advanced students open to assisting struggling or challenged students. Additionally, when available, ELL students will be provided with learning materials in their native language. Furthermore, if available, instructional aides will be provided to translate and assist ELL students with coursework. In development of software, the students will integrate multiple languages within the design of software.

5. Describe the interdepartmental articulation process for this course:

Students who successfully complete the introductory portion of this course will have laid a firm foundation in the most fundamental skills used in all areas of design, coding and testing of software. The aforementioned skills will help students successfully complete other courses within the Project Lead The Way curriculum for computer science. Hopefully as this program grows and expands we will be able to produce software that other departments have a need for and will aid in their instruction of students. Those same skills and problem solving techniques will help students succeed in courses across the curriculum at RHS and in life in general.

6. Describe how this course will integrate academic and vocational concepts, possibly through connecting activities. Describe how this course will address work-based learning/school to career concepts:

These are listed in each of the units and activities that are listed in section number 8.

7. Materials of Instruction (Note that materials of instruction for English language learners are required and should be listed below.)

- Textbook(s) and Core Reading(s):

Materials will be supplied through PowerPoints and written materials as supplied by the PLTW organization

- Supplemental Materials and Resources:

There will be extensive use of computers and Android tablet computers. And the use of the MIT App Inventor, MIT App Inventor and the computer language *Python®*

8. Course Outline

Introduction to Computer Science (ICS) is designed to be implemented as a half-year high school course or two nine-week PLTW Gateway modules. In each module, student teams create an Android® interface to solve a problem the team defines. Students learn fundamental computer science (CS) concepts using MIT App Inventor. The course aims to develop computational thinking and build student excitement. Several days in each module are targeted to build career awareness about computing skills in all fields and to improve students' cyber hygiene. The modules are sequential;

Tools

- MIT App Inventor
- MIT App Inventor
- *Python®*

Additional Skills and Knowledge

- Pair programming
- Project management
- Documentation
- Presentation
- Cyber hygiene
- Career and societal impact of computing

ICS 1: Mobile Computing (45 days)

Instructor's Preface

The goal of Unit 1 is to excite students about programming and build students' ability to break apart a problem and persistently build solutions in small steps. Student creativity, collaboration, and an iterative design process are emphasized. Students work with MIT App Inventor to create basic apps that rely on the concepts of event-driven programming, branching and iteration, variables, and abstraction – the building blocks of creating with code.

Student Preface

You will design and build apps to express yourself creatively. You may also choose to create something useful for people's needs. Many of the central skills of a programmer are useful in daily life:

- Breaking big problems into little ones
- Being persistent
- Building solutions in small steps
- Being creative
- Trying out your ideas

Look around you and appreciate the many luxuries of life that are created with code. Computer programs, created by software developers, are the instructions that make our smartphones, tablets, and computers smart. Are you ready to create intelligence?

Contents

Lesson 1.1 The Computing Revolution (18 days)

Instructor's Preface

The goal of this lesson is to introduce students to programming and to the impact of computing. With an introduction to pair programming and the software design process, students create original programs with MIT App Inventor. As part of the application-building process, students create audio and visual elements and examine how sound and images are represented in digital data. In app development, variables are used to introduce patterns that occur with iteration. Event-driven programming provides the iteration in this lesson, allowing students to inspect the value of variables after each iteration. The properties and procedures that belong to each component build student familiarity with the concepts of object-oriented programming.

Student Preface

In this lesson you will consider how computing is changing everything in our lives. You will be introduced to programming, the magic that makes it all happen. You will create original programs with MIT App Inventor, learning about pair programming and the software design process. You will use tools to create audio files and graphics that can be included in your apps. You will learn how computers handle digital data like images, sound, text, and numbers. You will give instructions to a computer by creating code using variables, functions, and operations like arithmetic.

Essential Questions

Q1. How has computing affected the world we live in?
Q2. Why is it advantageous to break a problem down into smaller pieces and build a solution incrementally?
Q3. How do computers represent the data in words, numbers, pictures, and sound?

Activities, Projects, and Problems – Summary and Goals

Activity 1.1.1 Computing Is Changing the World (2 days)

Students pick a grand challenge and consider how mobile computing, the Internet, Big Data, and simulation are contributing to solving that challenge.

Primary goals for the activity, project, or problem:
- *Establish the course as an exploration of the impact of computing*
- *Consider how computing will impact what is important to you*

Activity 1.1.2 Digital Doodle (2 days)

Students use MIT App Inventor (AI2) to create an app with a drawing canvas and its own camera control. The app allows users to draw on photos by dragging and tapping on the screen.

Primary goals for the activity, project, or problem:
- *Become familiar with the AI2 Designer and Blocks views*
- *Introduce AI2 components and user interface (UI)elements*
- *Use event handlers*

Activity 1.1.3 Count Me In (3 days)

Students create a mobile app with a counter operated by buttons and voice recognition. Students learn about the properties and events associated with AI2 components and are introduced to Agile development.

Primary goals for the activity, project, or problem:
- *Become familiar with storing, retrieving, and operating on string and numeric data*
- *Be able to identify and work with properties and events of labels and buttons*

Activity 1.1.4 Representing Music (2 days)
Students analyze digital and analog sound. Students use Audacity® software and a spectrum analyzer to create and analyze a digital recording of themselves.

Primary goals for the activity, project, or problem:
- *Use software to manipulate sound*
- *Describe how sound is represented in digital data*

Activity 1.1.5 Sound Decisions (3 days)

Students use an AI2 canvas to create a bouncing ball with sounds that depend on which side the ball bounces against.

Primary goals for the activity, project, or problem:
- *Use compound logic*

Activity 1.1.6 See-through (2 days)

Students use GIMP to create a sprite from an image. Representation and ownership of images are considered.

Primary goals for the activity, project, or problem:

- *Use software to manipulate images*
- *Describe the RGB and RGBA abstractions for representing images*
- *Describe fair use when rights are reserved and under Creative Commons*

Project 1.1.7 Sprite Smash (4 days)

Students create a game, Sprite Smash, in which a sprite pops up at random positions on the screen. The player scores points by tapping the sprite before it jumps to a new location. Students apply event handlers, procedures, global variables, and the Cartesian coordinate system.

Primary goals for the activity, project, or problem:

- *Use Cartesian x- and y-coordinates with AI2 properties*
- *Respond to internal and external events with event handlers*
- *Use a procedure to make code modular*
- *Increment and display global variables*

Lesson 1.2 Putting Together Pieces (16 days)

Instructor's Preface
Students continue to pair program and to explore the impact of the computing revolution as they learn how more complex programs are put together. Students build skills with collaboration tools and processes. They apply these collaboration skills while creating new MIT App Inventor projects. Students investigate community needs to identify an app they can develop to meet a real client's need.

Student Preface
In this lesson, you will learn how a complex program can be put together. You will learn how to break a big problem apart into manageable bite-sized pieces of success. Most software is created by teams that include dozens of people. How do they coordinate their work? You will learn about project planning and about tools that you can use to collaborate with others to create complex solutions to real problems. With a partner, you will conduct research and create the idea for an app to meet a real client's need.

Essential Questions

Q1. How is a complex piece of software organized?
Q2. How do teams plan and create complex solutions to a problem?

Activity 1.2.1 Picture Pool (5 days)

Students create an app in which a sprite slides around a canvas based on randomness, tablet tilt, flings, or taps. Abstracted procedures are provided and used to teach the concept of abstraction.

Primary goals for the activity, project, or problem:

- *Use velocity with Cartesian x- and y-coordinates*
- *Describe the purpose and concept of abstracting a procedure*
- *Introduce lists and iteration across lists*
- *Describe the role of argument values and return values*

Activity 1.2.2 Wikipedia That (2 days)

Students make meaning of a URL. They create an app in which the user can open side-by-side browsers to Google and Wikipedia using a text entry box and button.

Primary goals for the activity, project, or problem:

- *Understand the parts of a URL*
- *Practice constructing an app*

Project 1.2.3 Your Turn (2 days)

Students pick a task to complete. A crowdsourced document shared among teachers accumulates tasks in bite-sized pieces appropriate for students new to programming. Students may select from that list or branch out into new ground.

Primary goals for the activity, project, or problem:

- *Collaborate when programming*
- *Be persistent when programming*
- *Use documentation and other resources when programming*

Project 1.2.4 Decomposition (5 days)

Students pick a larger goal to complete, written as one or more user stories. Students break the user story into smaller tasks and complete a sprint toward their goal. A crowdsourced document shared among teachers accumulates successful sprints and their decomposition into tasks. Students may select from that list or branch out into new ground.

Primary goals for the activity, project, or problem:

- *Manage a project*
- *Decompose a problem*

Problem 1.2.5 What's Worth Making? (2 days)

Students interview a family member, a community member, and a school member while seeking a client for a mobile app. Students consider examples of how mobile and embedded computing are improving people's lives, and with what accompanying detriment. We're all engineers. What will you make?

Primary goals for the activity, project, or problem:

- *Collaborate effectively using team norms*
- *Identify needs that can be met with engineering*

Lesson 1.3 Collaborate to Solve Problems (11 days)

Instructor's Preface

Students examine how the computing revolution has affected collaboration and creativity. Ethical and safe behavior on the Internet is developed alongside opportunities and tools for collaboration over the Internet. In the culminating problem of the unit, students develop an app to meet a real client's need.

Student Preface

In this lesson, you will create an app for a real client's need. You will also improve your ability to safely and effectively use the Internet to collaborate with people. Whether creating a written product with a group or crowdsourcing data collection for a science experiment, people have new ways to work together using the Internet. No matter which career fields might interest you, computational thinking skills will benefit your career opportunities.

Essential Questions

Q1. How do I safely use the Internet?
Q2. How do people collaborate to create software applications?

Activities, Projects, and Problems – Summary and Goals

Activity 1.3.1 Digital Responsibility (2 days)

Students consider life as one big collaboration. Students reason about consequences for themselves and others in scenarios involving texting, creating and sharing pictures, posting to social media, and using email.

Primary goals for the activity, project, or problem:

- *Behave safely on the Internet and with digital communications*
- *Know some professional norms for digital communications*
- *Apply rules to respect intellectual property and collaborate effectively*

Project 1.3.2 Collaborative Writing (2 days)

Students within a school or in a pair of schools collaborate to create a product that includes text. The data will include both a text-encoded constrained-response data field and a prose data field. Examples could include a directory of local businesses or organizations, a curated list of websites about student interests, a biodiversity survey of plants and animals, or a compilation of student-written articles, comics, opinion pieces, and advertisements.

Primary goals for the activity, project, or problem:

- *Engage in collaborative, iterative writing*
- *Understand how computing has impacted the way we create with writing*
- *Understand how tools affect the way writing isrepresented and shared*

Activity 1.3.3 CS and IT Careers (2 days)

Students research and present about career opportunities in a field of their choice, focusing on the way in which CS and IT skills improve the opportunities in that career field.

Primary goals for the activity, project, or problem:

- *Describe career opportunities in CS and IT*
- *Describe how computing is impacting all fields*

Problem 1.3.4 Create an App for a Client (5 days)

Students develop an app to express creativity or to meet a need in a project growing out of the interviews in the previous lesson.

Primary goals for the activity, project, or problem:

- *Collaborate when programming*
- *Be persistent when programming*
- *Use documentation and other resources when programming*
- *Manage a project*
- *Decompose a problem*

ICS 2: Crowds and Clouds (45 days)

Instructor's Preface

In this unit, students explore the new opportunities for creativity and collaboration related to data. The two lessons focus on crowdsourcing and simulation as sources of data. In the first lesson, students build on previous experiences creating apps with MIT App Inventor. They modify apps to exchange data over the Web, culminating the first lesson of the unit by creating a crowdsourcing app. Ethical and safe behavior on the Internet is developed alongside an exploration of cybersecurity concepts. In the second lesson, text-based programming is introduced with *Python*®. Students simulate a game, generating data and transforming data. In the culminating problem of the lesson, students create an algorithm to play rock-paper-scissors and compete in a tournament. The competition motivates students to design an algorithm that can analyze data about the opponent's behavior.

Student Preface

This unit is about new opportunities for creativity and collaboration related to data. The two lessons focus on crowdsourcing and simulation as sources of data. In the first lesson, you will build on your previous experiences creating apps with MIT App Inventor. You will learn how to create an app that shares data over the Web. In the second lesson, you will learn how to program in a text-based language by creating simulations of games in *Python*®.

Contents

Lesson 2.1 Coding for the Crowd (21 days)

Instructor's Preface

The goal of this lesson is to reinforce students' understanding and enthusiasm for computing as a powerful tool for collaboration. Activities explore how information is presented and exchanged on the Web. Building on their new understanding of the Web, students develop an app that transmits and receives data from a Web service through an application programming interface (API). In the final problem, students develop an app to crowdsource data collection on a topic of their interest and then analyze the data.

Student Preface

In this lesson, you will consider computing as a powerful tool for collaboration. In the first activity, you learn how information is presented and exchanged on the Web. You then explore how an app can use the Web to share data among many devices: data like social posts, "likes," leader boards, friend lists, and shared images or comments. In the final problem,you will create a crowdsource data collection app related to a topic of your choice.

Essential Questions

Q1. How do apps share data across devices through the Internet to let users to interact?
Q2. What data are you contributing via your interactions on the Web andthrough apps, and to whom are you contributing the data?
Q3. What new phenomena are being created when many users are contributing to a data set?
Activities, Projects, and Problems – Summary and Goals

Activity 2.1.1 What Is a Web Page? (5 days)

Students explore basic HTML and CSS, the languages of the Web. Students manipulate a locally stored Web page, adding elements and modifying the background color, reinforcing hexadecimal RGB color representation.

Primary goals for the activity, project, or problem:
- *Understand URLs and the client-server relationship*
- *Understand the purpose of HTML and CSS*
- *Generalize that data sit atop many layers of abstraction, with zeros and ones at one low-level layer*

Activity 2.1.2 Web API Service (2 days)

Students learn how to use an application programming interface (API) to send commands to a Web server over the Web. By using an interface other than a browser, they learn about GET and POST requests over the Web's HTTP protocol. They post a phrase to a class "wall" on a Web server, interpret data from the Web server written in JavaScript Object Notation (JSON), and vote for their favorite phrases.

Primary goals for the activity, project, or problem:
- *Understand how apps send and receive data over the Web*
- *Deepen understanding of Web protocols and URLs*

Activity 2.1.3 App for a Web Service (3 days)

Students use MIT App Inventor to create a simple app to allow a user to send and receive API data over the Web. They automate the sending of data in a cybersecurity
challenge.

Primary goals for the activity, project, or problem:

- *Practice constructing an app*
- *Understand how to exchange data with a Webservice*

Activity 2.1.4 Collaborative Data (2 days)

Student use a Google sheet to share data about themselves with the class. Patterns are observed and compared between two groups. Students discuss personally identifiable information (PII) and safe/common/legal practices regarding PII. Students crowdsource the collection of data for questions of interest to them and consider the effectiveness of measures to de-identify and analyze the data.

Primary goals for the activity, project, or problem:

- *Understand how the production and collection of data can be crowdsourced*
- *Describe how computation has changed science*
- *Compare center, spread, and shape for two distributions*

Problem 2.1.5 Create a Crowdsourcing App (6 days)

Students develop an app that shares data across multiple users. Students have the option to embellish and further develop their app from Problem 1.3.4, now using the power of crowdsourced data.
Primary goals for the activity, project, or problem:

- *Develop an app for a client that leverages the power of shared data*
- *Manage a project*
- *Decompose a problem*

Problem 2.1.6 Authentic Audiences (3 days)
Students reflect on their work from Problem 1.3.4. Teams present their process and product to the class, to the client, or to an end user.

Primary goals for the activity, project, or problem:

- *Reflect on and improve a development process*
- *Present a product*

Lesson 2.2 Cracking the Code (24 days)

Instructor's Preface

The goal of this lesson is for students to become comfortable implementing algorithms using conditionals and loops in *Python*® and to generalize algorithmic structures from corresponding MIT App Inventor and *Python*® code. Students create a game simulation, learning about functions, arguments, and return values. Students generalize from this simulation to learn about model abstraction and the impact that simulation and data are having across all career fields. Students then apply their *Python*® skills to compete in a rock-paper-scissors game, developing functions to implement a complex strategy that attempts to detect their opponent's strategy.

Student Preface

In this lesson you will learn how to create algorithms with *Python*®, a text based programming language. You will create a game simulation and explore how simulation is affecting all career fields. You will find that you can transfer what you learned in MIT App Inventor® to text-based languages like *Python*®. Your class will hold a round-robin tournament in which you will take advantage of computing power to predict another player's strategy in the rock-paper-scissors game. Through programming these games, you will learn the central principles of algorithms, the recipes that control what the computer does.

Essential Questions

Q1. How are algorithms used to solve common problems?
Q2. How are functions and abstraction used to handle complexity?
Q3. How are data and simulation affecting career fields?

Activities, Projects, and Problems – Summary and Goals

Activity 2.2.1 Winning Distribution (1 day)

Students collect data about outcomes in Ezee, a game in which outcomes are random and players try to get 14 of a kind.

Primary goals for the activity, project, or problem:
- *Consider questions that can only be answered by considering a distribution of values for a variable*
- *Consider questions that are most easily answeredwith simulation*

Activity 2.2.2 Exploring *Python*® (2 days)

Students explore a *Python*® development environment and become familiar with a code editor and an interactive command line.

Primary goals for the activity, project, or problem:
- *Increase comfort with text-based programming*
- *Distinguish data types*

Activity 2.2.3 *Python*® Functions (3 days)

Students define and call functions with arguments to accomplish simple mathematical tasks.

Primary goals for the activity, project, or problem:
- *Increase comfort with text-based programming*
- *Be able to define and call functions with arguments*

Activity 2.2.4 Double Meanings (2 days)

Students compare the meaning of the terms “variable,” “function,” and “equal” in the contexts of mathematics and computer programming languages.

Primary goals for the activity, project, or problem:

- *Dispel common misconceptions about computer science concepts*
- *Reinforce core concepts in mathematics*

Activity 2.2.5 Looping Patterns (4 days)

Students learn three patterns for loops: accumulation, aggregations, and finding the maximum or minimum in a set. For each pattern, students study an example, complete an example, and then create their own code.

Primary goals for the activity, project, or problem:

- *Understand how functions use arguments and return values*
- *Recognize and create iteration patterns*

Project 2.2.6 Simulation Game (5 days)

Students create a sequence of *Python*® functions to simulate a single game of Ezee, which they played at the beginning of the lesson.

Primary goals for the activity, project, or problem:

- *Understand how functions use arguments and return values*
- *Gain confidence and expertise with patterns involving iteration*
- *Understand how complex problems can be solved by creating modular components that build upon each other*

Activity 2.2.7 Data and Simulation Everywhere (2 days)

Students explore a distribution resulting from a Monte Carlo simulation and identify which details of a phenomenon are parameterized and which details are abstracted away by a model. Students research the impact of modeling and simulation in a career field of their choice.

Primary goals for the activity, project, or problem:

- *Describe the impact of simulation and modeling in various career fields*
- *Understand modeling as a type of abstraction*

Problem 2.2.8 Strategy Game PS Rock (5 days)

Students create an algorithm to analyze a competitor's history in rock-paper-scissors and predict the competitor's next move. Students implement their algorithm in *Python*® and compete in a round-robin tournament.

Primary goals for the activity, project, or problem:

- *Create a complex algorithm*
- *Implement an algorithm in Python®*