James Logan High School
Computer Science Principles
Course Syllabus

Instructor: Ms. Banther

Project Lead the Way courses are introductory courses in computer science. These courses have a weighted grade system. Those who wish to receive college credits will have to receive a B or higher in the class.

Welcome! You will now begin your training in Computer Science. AP Computer Science Principles is an Advanced Placement course in which you may receive college or university credit. In order to receive college credit you must take a comprehensive test, which will cover the material covered in this class. To do well on this test, it is essential that we complete this class by the end of April. We will then have two weeks to review for the AP exam. Plan on at least ½ hour of work outside of class per day.

Texts: Blown to Bits, Hal Abelson, Ken Ledeen & Harry Lewis 2008 edition and PLTW(Project Lead the Way) Computer Science Lessons. You will be provided these resources electronically for you to use in class and as reference at home.

Homework: Homework may be assigned daily. You may be given 5-10 pages of reading per day and/or problems to solve on paper. You are encouraged to work with others on your homework in order to check your accuracy. Working together should be a mutual effort. Avoid situations where you will be just recording the answers, as this will hurt you later. Find a partner who can accept your help and give you help without doing the work for you. All homework assignments are due at the beginning of the period, although not all of them will be checked.

Grading: We will utilize the Formative/ Summative grading policy. The Formative part (HW, Projects and Quizzes) will make up 30% of your grade and Summative (Tests, Summative Projects and Final) will be the remaining 70%. There will not be separate categories just **Formative and Summative.** The grading scale is the standard 90%-100% A, 80%-89% B... percent table.

Make-Up and Late work: It is your responsibility to make up your work when you are absent. Tests and homework must be made up within a reasonable amount of time upon your return from an excused absence. Some quizzes may not be made up and therefore will not count against you if your absence is excused. Late homework is accepted for full credit 1 week after the due date and for up to half credit up to the end of the semester.

Groups: For the first semester, most of you will work in groups of two or three in class. Choose your partner carefully.

Academic Dishonesty will result in zero for all parties involved. Group work is acceptable copying is not.

Course Description
Students work in teams to develop computational thinking and solve problems. Structured activities progress to open-ended projects and problems that require planning, documentation, communication, and other professional skills. Problems aim for ground-level entry with no ceiling: all students can successfully engage the problems while students showing greater achievement are challenged to work further. There are five primary course objectives.

- To develop problem solving and computational thinking skills
- To generate excitement about the field of computing
- To introduce computational tools that foster creativity
- To build awareness of career opportunities in all fields for people with computational skills
- To consider issues raised by the present and future societal impact of computing

CS Principles: Computational Thinking Practices and Big Ideas
Students develop the six Computational Thinking Practices and achieve enduring understandings of the seven Big Ideas by collaborating to create solutions to problems. They evaluate and communicate solutions and connect computing to all areas of work and society. In considering any problem, students build the habit of considering five ways in which computing can be applied to a problem.

- Express ideas with creativity

Use tools to create and remix artifacts including sound, images, video, and text, and visualizations. Understand how such data are represented digitally.

- Safely and effectively use the Internet

Use the Internet for collaboration, crowd-sourcing, research, publication, social networking, and economic participation. Understand principles of cyber hygiene.

- Collect, visualize, analyze, and communicate data

Formulate problems so that computation can help solve them, logically organize data, and generalize patterns in data.

- Model and simulate

Identify abstractions in a model and understand, predict, and communicate phenomena with simulation.

- Create and improve algorithms and automate

Describe and improve an algorithmic procedure. Create software for networked and embedded computing and physical automation.

Course Style
Each lesson is framed by a project or problem. Students begin each lesson by considering the problem and the lesson's essential questions, and they reflect throughout the lesson on the essential questions. The knowledge and skills necessary for students to successfully engage the problem are taught during activities guided by step-by-step instructions, instructional videos, teacher presentations, and unplugged activities.

Students complete the vast majority of the work in the course with a partner, typically changing partners each lesson.

Texts and Learning Resources
Abelson et al. (2008). *Blown to Bits*. Addison Wesley.
Passages are used in specific assignments.
Activities, projects, and problems and supporting resources are provided to the student in offline and online formats. The offline format includes Word documents, PowerPoint presentations, MP4 instructional videos, and zip files of source code and data. These resources are also available through a web-based interface with the Instructure Canvas learning management system, with hyperlinks to additional external resources.

Lab Computing Resources and Coding Opportunities
Lab resources are sufficient to provide each pair of students with two computers meeting PLTW computer specifications and one Android device with WiFi connectivity for live testing in MIT App Inventor. SSH access is provided to a Linux environment for learning about the Internet, the Web, and cybersecurity. The syllabus calls for students to be working with code on a computer during 87 days plus additional time for the *Create* performance task. The course software installation list indicates software provided to the student, including Canopy Python, NetLogo, XaoS, MEGA, and tools for SSH, FTP, and hex inspection.

Computer Languages Used
During the unit on algorithms, students are introduced to programming with Scratch™ and MIT App Inventor and then use *Python®. Python* is the primary language used in Unit 1 and in the course overall. During the unit on the Internet, students use *Python* and also work in a Linux environment manipulating HTML, CSS, JavaScript, PHP, and SQL. During the unit on data, students use *Python* and Microsoft® Excel®. During the unit on simulation, students use *Python* and NetLogo.

Learning Objectives
The course follows the Understanding by Design model in which each lesson is designed to produce evidence that students have achieved specific course objectives. Course objectives include all CS Principles learning objectives. Course materials explicitly show alignment to objectives at the lesson level and at the activity level, including PLTW understanding, skills, and knowledge objectives; CS Principles learning objectives and essential knowledge; and objectives specified in NGSS, CCSS, CSTA, and other standards. There are many opportunities to meet the learning objectives within each of the big ideas; several are described below. In all of these examples, students work in a team to combine team members' perspectives, skills, and knowledge to address a complex problem. This team approach gives students opportunities to describe, explain, and justify the design and appropriateness of their computational choices, to analyze and describe computational artifacts and the behavior of programs, to communicate the meaning of knowledge discovered in data, and to use simulations to understand, predict and communicate about natural phenomena.

Example activities, projects, and problems
Each activity, project, or problem provides opportunities to meet learning objectives within several big ideas, often focused primarily within one big idea. Students engage in all six computational thinking practices: Connecting Computing [P1], Creating Computational Artifacts [P2], Abstracting [P3], Analyzing Problems and Artifacts [P4], Communicating [P5], and Collaborating [P6].

Big Idea 1: Creativity
 In Problem 1.2.6 Designing an App, student teams create an app for a mobile device using MIT App Inventor. Teams pair program and apply concepts of Agile development. Students present their creations and reflect in writing on their development process and on their collaboration.

Big Idea 2: Abstraction
 In the unplugged Activity 1.4.1 Procedural Abstraction, students act out the instantiation of ping pong balls and implement methods for drawing colored shapes on the surface of the ping pong balls. Students generalize to consider how classes and methods abstract away details.

 In Activity 4.1.3 Introducing Simulation, students explore hypotheses about a wolf-sheep-grass ecosystem abstracted by a simulation in NetLogo. Continuing this work in Activity 4.1.4 Varying Simulation Parameters, students analyze the wolf sheep predation model using Monte Carlo methods and use parallel processors to automate exploration of the behavior of the model across a range of parameter values. In another model using moths, students analyze data to determine the characteristics of moth flight that reproduce real motion captured in a video.

Big Idea 3: Data and Information
 In Project 3.2.6 Genomic Data, each student team picks a human protein, retrieves a published DNA sequence from the National Center for Biotechnical Information, and executes a BLAST search for similar DNA sequences in the Center's database. Students use MEGA, a professional biologist's tool, to create a phylogenetic tree visualizing the similarities in the genetic data across organisms. .

Big Idea 4: Algorithms
 In Activity 2.3.2 Security by Encryption, student teams analyze different algorithms that decide whether a number is a prime. Students use Python's timeit library to empirically compare the algorithms and analyze how the time efficiency of the algorithms depends on the length of the data input.

Big Idea 5: Programming
 In Activity 1.4.5 Image Algorithms, student teams design and implement an algorithm to create derivative images from all images in a folder. Students use the numpy library for scientific computing to code pixel-level analysis and manipulation. Students also use the Python Imaging Library to code image manipulations at a higher level of abstraction. Students explain abstractions they use and explain how their program functions.

Big Idea 6: Internet
 In Activity 2.1.3 Protocols and Bandwidth, students act out the TCP/IP protocol and recursive DNS lookup. Student use a Linux environment provided by PLTW to use various tools for understanding DNS, routing, latency, and bandwidth. Tools used include ping, the Domain Information Groper, ifconfig, nslookup, and tracepath.

Big Idea 7: Global Impact
 In Activity 2.1.1 Rise of the Internet, student teams compare Internet connectivity between groups of countries over time. Students then explore the global impact of the Internet, selecting one topic from

choices including social interaction, retail business, scientific research, and governance. Each team identifies reliable sources of information and prepares a two- to five-minute presentation on their topic.

 In Activity 2.3.3 Security and Liberty, students read, discuss, and write about data generated by and about citizens and consider impact on privacy, liberty, and law enforcement. Students explore what data sets are available from the government and describe the content of one of them with consideration of the impact the shared data has on society. Students read one or more ACM Tech News articles and write about beneficial and harmful effects of a computing innovation.

Five to ten days (45-60 minutes each) are allotted for each of the projects and problems that offer practice opportunities for the *Create* task. One or two days are allotted for each of the activities that offer *Explore* task practice opportunities. Students are provided with 8 hours of class time to complete the *Explore* performance task and 12 hours of class time to complete the *Create* performance task using the College Board Performance Task descriptions and rubrics. Categorized as Problem 4.2.5, these 20 hours are distributed throughout the course, with roughly 5, 10, and 5 hours devoted in Units 2, 3, and 4, respectively.

Additional Assessment
Students are encouraged but not required to take the AP CS Principles exam. Students are required to complete the 80-minute PLTW end of course exam. A handful of check-for-understanding quizzes are used to assess progress during course activities.
Student work on the projects and problems forms the primary evidence of student achievement. PLTW rubrics are provided for student deliverables including presentations, writing, collaborations, project documentation, and software. The College Board Performance Assessment Rubrics are also used during the practice opportunities, focusing on different elements of the rubrics at each of the opportunities.

Syllabus at a Glance
Unit 1: Algorithms, Graphics, and Graphical User Interfaces (68 days, 9 additional days as time permits)
Comprising the first half of the course, Unit 1 emphasizes programming, creativity, algorithmic thinking, and abstraction. Students work with Scratch, App Inventor, and *Python* programming languages to tell graphical stories, publish games and Android applications, and explore various development environments and programming techniques. Students create original code and read and modify code provided from other sources. An Agile software development process is emphasized, and personal, professional, and collaborative skills take center stage. Students debate policy questions about the ownership and control of digital data and examine the implications for creative industries and consumers. In this unit students begin their exploration of career paths tied to computing.
Lesson 1.1 Algorithms and Agile Development (15 days)
Lesson 1.2 Mobile App Design (15 days)
Lesson 1.3 Algorithms in *Python* (20 days)
Lesson 1.4 Images and Object-Oriented Libraries (17 days)
Lesson 1.5 GUIs in *Python* (10 days, optional)

Unit 2: The Internet (30 days)
The goal of Unit 2 is for students to have a more concrete understanding of the Internet as a set of computers exchanging bits and the implications of these exchanges. Students use PHP and SQL to

structure and access a database hosted on a remote server, learn how HTML and CSS direct the client computer to render a page, and experiment with JavaScript™ programming language to provide dynamic content. The focus of the unit is on the protocols that allow the Internet to function securely as it delivers social media and eCommerce content. Students work briefly in the various web languages to understand how the languages work together to deliver this content. The history and workings of the Internet are explored, and issues of security, privacy, and democracy are considered. Practical cybersecurity hygiene is included. Career paths in cybersecurity, web development, and information technology are highlighted.
Lesson 2.1 The Internet and the Web (9 days)
Lesson 2.2 Shopping and Social on the Web (13 days)
Lesson 2.3 Security and Cryptography (8 days)

Unit 3: Raining Reigning Data (30 days)
The goal of Unit 3 is for students to see the availability of large-scale data collection and analysis in every area they can imagine. Students examine very large data sets tied to themselves as well as to areas of work and society. They learn a variety of data visualization techniques and work to recognize opportunities to apply algorithmic thinking and automation when considering questions that have answers embedded in data. The complexity of the data sets, visualizations, and analysis increases in the second lesson of the unit, challenging students to generalize concepts developed in the first lesson.
Lesson 3.1 Visualizing Data (14 days)
Lesson 3.2 Discovering Knowledge from Data (16 days)

Unit 4: Intelligent Behavior (24 days)
In Unit 4 the emergence of intelligent behavior is explored from two distinct approaches: from human crowd sourcing of data and from separate algorithmic agents working in parallel. The goal is to galvanize the connections among computing concepts and between computing and society. The first lesson explores the hardware layer of computing, working from discrete components to integrated circuits. The exponential advancement of electronics, low on the ladder of abstraction, is connected to advancements at the highest levels on the ladder of abstraction, where artificial intelligence and simulation and modeling are impacting all fields. In the concluding lesson, students identify problems and questions that can be addressed with computer simulation, incorporating agent-based modeling. Students are challenged to explore the assumptions and parameters built into several simulations and to attach meaning to the results. Having explored a few applications of intelligent behavior emerging from algorithmic components, students reflect on the current and future state of artificial intelligence.
Lesson 4.1 Moore's Law and Modeling (12 days)
Lesson 4.2 Intelligent Agents (12 days)

Through-Course and End of Course Assessment (21-22 hours)
Performance Task *Create* (12 hours)
Performance Task *Explore* (8 hours)
PLTW Assessment PLTW End-of-Course Exam (Two 40-minute sessions)
AP Assessment AP End-of-Course Exam (2 hours, optional)