OMAHA NORTH HIGH MAGNET SCHOOL

Honors Computer Science and Software Engineering
Course Syllabus

Instructor: Mr. Randall Henderson
E-mail: Randall.Henderson@ops.org
Office Phone Number: 531-299-8922
Plan Periods: TBD
Office/Classroom: Room 143
Best times to contact: Via email – my plan period Via in-person – after-school

Course Description:

Computer Science Principles (CSP) curriculum is a full year, rigorous, entry-level course that introduces high school students to the foundations of modern computing. The course covers a broad range of foundational topics such as programming, algorithms, the Internet, big data, digital privacy and security, and the societal impacts of computing.

First semester will cover the first 3 Units, and second semester will continue with Unit 4 with the option of taking the AP test in May. This syllabus covers first semester.

Course Snapshot

To the right is a snapshot of the course. The course contains five core units of study, with a sixth unit devoted almost exclusively to students working on their AP Performance Task (PT) projects. It is NOT required for students to take the AP test.

Each unit has one or two “chapters” of related lessons that usually conclude with project or summative assessment. A timeline showing a typical school year is shown to give a rough estimate of pacing. Note: for AP credit, the performance task submission deadline is the end of April, and the written AP Exam is October 15th,

AP Endorsed

This curriculum is recognized by the College Board as an endorsed AP® Computer Science Principles class (AP CSP). This endorsement affirms that all components of Code.org CSP’s offerings are aligned to the AP Curriculum Framework standards and the AP CSP assessment.
Students can choose to either take this class one or two semester and are not required to take the AP test as a part of the class.

**Curriculum Overview and Goals**

Computing affects almost all aspects of modern life and all students deserve a computing education that prepares them to pursue the wide array of intellectual and career opportunities that computing has made possible.

This course is not a tour of current events and technologies. Rather, this course seeks to provide students with a “future proof” foundation in computing principles so that they are adequately prepared with both the knowledge and skills to live and meaningfully participate in our increasingly digital society, economy, and culture.

The Internet and Innovation provide a narrative arc for the course, a thread connecting all the units. The course starts with learning about what is involved in sending a single bit of information from one place to another and ends with students considering the implications of a computing innovation of their own design.

Along the way students learn:

- How the Internet works and its impacts on society.
- How to program and rapidly prototype small JavaScript applications both to solve problems and to satisfy personal curiosity.
- How to collect, analyze and visualize data to gain insight and knowledge.
- How to evaluate the beneficial and harmful effects to people and society brought on by computing innovations.

The curriculum assumes no prior knowledge of computing and is written to support both students and teachers who are new to the discipline. Activities are designed and structured in such a way that students with diverse learning needs have space to find their voice and to express their thoughts and opinions. The activities, videos, and computing tools in the curriculum strive to have a broad appeal and to be accessible to a student body diverse in background, gender, race, prior knowledge of computing, and personal interests.

Broadening student participation in computer science is a national goal, and effectively an issue of social justice. Fancy tools and motivational marketing messages only get you so far. We believe that the real key to attracting students to computer science and then sustaining that growth has as much to do with the teacher in the classroom as it does with anything else. The real “access” students need to computing is an opportunity to legitimately and meaningfully participate in every lesson regardless of the student’s background or prior experience in computing coming into the course. For example, the course begins with material that is challenging but typically unfamiliar even to students who have some prior experience or knowledge of computer science. Students should not feel intimidated that others in the class are starting with a leg up on the material.

**Who Should Take This Course?**

The curriculum itself does not assume any prior knowledge of computing concepts before entering the course. It is intended to be suitable as a first course in computing though students with a variety of backgrounds and prior experiences will also find the course engaging and with plenty of challenges. While it is increasingly likely that students entering this AP course in high school will have had some prior experience in computer science (particularly with programming), that experience is equally likely to be highly varied both in quantity and quality. It is for this reason that the course does not start with programming, but instead with
material that is much more likely to put all students on a level playing field for the first few weeks of class. Read more about this below in the description of Unit 1.

**Unit Structure: Units, Chapters, Lessons**

While the layout of units appears to be modular, the units of study are loosely scaffolded, and sequenced to build students’ skills and knowledge toward the Enduring Understandings of the CSP Course Framework. The lessons for each unit assume that students have the knowledge and skills obtained in the previous units. There are also many thematic connections that can be made between and among lessons and units.

Each unit attempts to “tell a story” about a topic in computing from a more primitive beginning to a more complex end. The lessons in each unit are grouped into one or two chapters of a few weeks’ worth of lessons whose content is related or connected in some way. The course snapshot on the first page shows the chapters for each unit. Each lesson is intended to be a complete thought that takes the student from some motivational question or premise to an activity that builds skills and knowledge toward some learning objective(s).

Each unit contains at least one summative assessment, or project. Sometimes these come mid-unit, and sometimes they come closer to the end.

**Lesson Structure and Philosophy**

Lessons are designed to be student-centered and to engage students with inquiry-based and concept-discovery activities. The course does not require the teacher to lecture or present on computer science topics if they do not want to. Direct instruction, where necessary, is built into our tools and videos.

Another goal of each lesson is to provide more resources, supports, and activities than a teacher could (or should) use in one lesson. The teacher plays a large role making choices and ensuring that the activities, inquiry, and reflection are engaging and appropriate for their students, as well as assessing student learning.

Most lessons have the following structure:

- A warm-up activity to activate prior knowledge and/or present a thought-provoking problem
- An activity that varies but is typically one of 5 possible types:
  - Unplugged concept invention and problem-solving scenarios
  - Using a computational widget to discover concepts more deeply
  - Using external computational tools (such as spreadsheets, or presentation tools)
  - Programming in App Lab
  - Research and communication (Research / Writing / Reflection / Presentation)
- A wrap-up activity or reflection to pull together the core concepts of the lesson

**Technical Requirements**

All the course tools and resources (lesson plans, teacher dashboard, videos, student tools, programming environment, etc.) are online and accessible through a web browser. You will have access to a Mac laptop computer during class.

While the course features many “unplugged” activities away from the computer, daily use of a computer is assumed. It is not required that students have access to computers at home, but because almost all the
materials are online, students with access to computers outside of class and at home will have access to much of their coursework outside of class.

**Computational Tools, Resources and Materials**

The Code.org CSP curriculum includes almost all resources teachers need to teach the course including:

**Lesson Plans**
- Instructional guides for every lesson
- Activity Guides and handouts for students
- Formative and summative assessments
- **Exemplars, rubrics, and teacher dashboard**

**Videos**
- Student videos - including tutorials, instructional and inspirational videos
- Teacher videos - including lesson supports and pedagogical tips and tricks

**Computational Tools**
- Widgets and simulators for exploring individual computing concepts
- Internet Simulator - Code.org’s tool for investigating the various “layers” of the internet
- App Lab - Code.org’s JavaScript programming environment for making apps

A few lessons call for typical classroom supplies and manipulatives such as poster paper and markers, as well as additional materials like Dixie cups, string, playing cards, a handful of Lego blocks, etc. In most cases there are alternatives to these materials if necessary.

**Suggested Text:**


This course does not require or follow a textbook. Blown to Bits is a book that can be accessed online free of cost. Many of its chapters are excellent supplemental reading for our course, especially for material in Units 1, 2 and 4. We refer to chapters as supplemental reading in lesson plans as appropriate.

**AP® Assessment**

The AP Assessment consists of a 74-question multiple choice exam and two “through-course” assessments called the AP Performance Tasks (PTs). The tasks can be found in the official AP CS Principles Exam and Course Description.

- Create Performance Task (p. 108)
- Explore Performance Task (p. 111)

**Assessments in the Curriculum**

The course provides several assessment types and opportunities.
Summative Assessments:

The curriculum contains two types of summative assessments.

1. Fixed Response Assessments

   After a group of concepts has been adequately covered - typically this means every 5-8 lessons (roughly every few weeks) - a fixed response assessment with items such as multiple choice, matching, choose two, short answer, etc. appears in the curriculum.

2. Practice Performance Task Assessments

   Each unit contains at least one project. These projects are smaller in scope, contextualized to the unit of study.

Worksheets and Activity Guides

- Many lessons contain worksheets or activity guides that ask students to write, answer questions, and respond to prompts (Answer keys provided) that could be used as formative assessment.

Unit Overviews

What follows are more in-depth descriptions of each unit of study which explain the topics covered and what students will be doing. Each unit also highlights a particular lesson, project or assignment of interest, explaining what students do and showing which learning objectives and computational thinking practices that particular assignment addresses. In the unit descriptions we also reference the Enduring Understandings from the CSP Framework with bolded text and a parenthetical number. For example, you might see: people write programs to execute algorithms (5.1). See the CSP Framework document for listings of all the Enduring Understandings.

Unit 1: The Internet

This unit explores the technical challenges and questions that arise from the need to represent digital information in computers and transfer it between people and computational devices.

Topics include: the digital representation of information - especially, numbers, text, and communication protocols. The first unit of this course purposefully addresses material that is fundamental to computing but with which many students, even those with computers at home or who have some prior experience with programming, are unfamiliar. This levels the playing field for participation and engagement right from the beginning of the course.

Unit 2: Digital Information

This unit further explores the ways that digital information is encoded, represented and manipulated. In this unit students will look at and generate data, clean it, manipulate it, and create and use visualizations to identify patterns and trends. Students will use a variety of tools including Code.org widgets and external data manipulation and visualization tools (such as Excel or Google Sheets).

Unit 3: Algorithms and Programming

This unit introduces students to programming in the JavaScript language and creating small applications (apps) that live on the web. This introduction places a heavy emphasis on understanding general principles of computer programming and revealing those things that are universally applicable to any programming language.
Unit 4: Big Data and Privacy

The data rich world we live in also introduces many complex questions related to public policy, law, ethics and societal impact. In many ways this unit acts as a unit on current events. It is highly likely that there will be something related to big data, privacy and security going on in the news at any point in time. The major goals of the unit are 1) for students to develop a well-rounded and balanced view about data in the world around them and both the positive and negative effects of it and 2) to understand the basics of how and why modern encryption works.

Unit 5: Building Apps

This unit continues to develop students’ ability to program in the JavaScript language, using Code.org’s App Lab environment to create a series of small applications (apps) that live on the web, each highlighting a core concept of programming. In this unit students transition to creating event-driven apps. The unit assumes that students have learned the concepts and skills from Unit 3, namely: writing and using functions, using simple repeat loops, being able to read documentation, collaborating, and using the Code Studio environment with App Lab.

Unit 6 - Performance Tasks

In Units 1-5 students learned and practiced the skills and content they needed to know in order to succeed on the AP CSP Performance Tasks. Still, a certain level of guidance during the PT development process is not only recommended, but vital. For example, coaching students early on helps them clarify their ideas and/or approaches to the PTs. The only real new instruction in this unit is to shore up understandings for the Explore Performance task related to finding, evaluating and citing sources (7.5).