Contents

Computing Curriculum **1**

- License 1
- Advanced Information Systems 1
  - Contact 1
- Collaborate 1
- Preface 1
  - Problem 1
  - Mission 1
  - Rationale 2
  - Values 2
  - Critical Consideration 2
  - Assumptions 3
  - High-School STEM Education 3
  - Domains 3
- Primer Seminars 4
  - Professional Collaboration and Consultation 5
  - Digital Literacy 5
- Year 1 6
  - Artifacts of Learning (examples) 6
  - Learning Objectives 6
- Year 2 10
  - Artifacts of Learning (examples) 11
  - Learning Objectives 11
- Year 3 13
  - Artifacts of Learning (examples) 13
  - Learning Objectives 13
- Year 4 15
  - Artifacts of Learning (examples) 15
  - Learning Objectives 15
- Authentic Learning & Assessment 16
  - Note 16
  - Research-Based Pedagogy 16
  - Examples 16
- Anchor Texts 17
  - Text-Based Computer Languages 17
  - Linux and Shell Scripting 17
  - Visual Languages 17
  - Hardware and Electronic Engineering 17
  - Raspberry Pi 17
  - Post-Secondary Transition 17

Miscellaneous 18
Prospective Curriculum Technologies 18
References 19
Indices and tables **19**

Computing Curriculum

Contents:

License

Advanced Information Systems

Contact

Rik Goldman

Chair, Technology Department

English Teacher

Chelsea School

rgoldman@chelseaschool.edu

@9_while_9 (Twitter)

[More contact info - placeholder]

Collaborate

This documentation was imagined as a product of collaboration. To facilitate collaboration, the most current draft is on github at https://github.com/ghoulmann/cs-curriculum (scroll down for the README and important information for collaborators).

Collaborating will involve making a github account, forking the repository, and having git installed on your internet-aware computer. The document is written using reStructuredText, a very handy markup syntax. I edit using an application called ReText which offers document preview; there are others around and your sure to find one you can work with.

When you begin editing, please add your name to the LICENCE document so you receive the credit you will certainly deserve.

Rik Goldman

rgoldman@chelseaschool.edu

@9_while_9 (Twitter)

Preface

Problem

It is not enough to equip students to access, consume, and utilize digital tools.

Mission

Empower students to participate in the production of a intentionally designed, digital world through systematic and computational thinking based on authentic instruction in the complimentary fields of Literacy, Computer Science, Electronic Engineering, and Information Technology.

A successful implementation ensures that all participating students

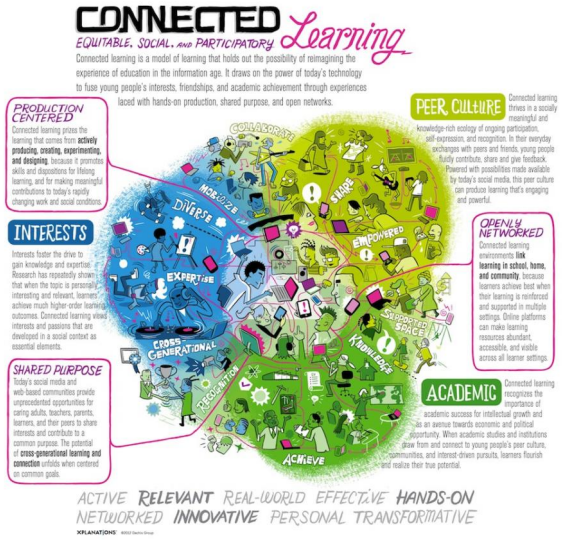
- Are competent, confident, and creative users of information technology
- Can critically evaluate and apply information technology (including new or unfamiliar technologies) responsibily, collaboratively, and effectively to solve problems
- Can analyze problems in computational terms, and have repeated practical experience of writing computer programs in order to solve them
- Can understand and apply the fundamental principles of computer science, including logic, algrorithms, data representation, and networks
- Can critically articulate the individual, cultural, and societal impacts of digital technology and know how to stay safe, exploit opportunities, and manage risks (*Draft ICT Programme of Study*)

Rationale

This program encourages innovation, collaboration, and resourcefulness as it develops and requires logical thinking and precision. Students apply underlying principles to understand, manipulate real-world systems and to create purposeful and usable artifacts. Underlying principles, authentic practice, innovation, and invention make it both rigorous and creative. (*Draft ICT Programme of Study*)

Values

- Equity
- Contructivist pedagogy
- Literacy (reading and writing) across the curriculum
- citizenship
- Authentic learning and assessment [narrowly defined; distinguished from project & performance-based assessment]
- Differentiated Instruction
- Collaboration
- Networked learning (Ito 74) [1]



Critical Consideration

- Underrepresentation of women in STEM fields
- Access to resources

- Reluctant readers and readers with language-based learning differences

Assumptions

Computer Science and Electronics Engineering are STEM disciplines (science, technology, engineering, math). Computer Science and Information Technology are complementary subjects (CAS 4).

As contrasted by the Computing at School Working Group in 2012,

> Computer Science is a *discipline* that seeks to understand and explore the world around us, both natural and artificial, in computational terms. Computer science is particularly, but by no means exclusively, concerned with the study, design, and implementation of computer systems, and understanding the principles underlying these designs.
>
> Information Technology deals with the purposeful application of computer systems to solve real-world problems, including issues such as the identification of business needs, the specification and installation of hardware and software, and the evaluation of useability. It the productive, creative and explorative use of technology. (CAS 5)

High-School STEM Education

From *Maryland State STEM Standards of Practice Framework*:

> STEM education is an approach to teaching and learning that integrates the content and skills of science, technology, engineering, and mathematics. STEM Standards of Practice guide STEM instruction by defining the combination of behaviors, integrated with STEM content, which are expected of a proficient STEM student. These behaviors include engagement in inquiry, logical reasoning, collaboration, and investigation. The goal of STEM education is to prepare students for post-secondary study and the 21st century workforce.
>
> STEM education removes the artificial barriers that isolate content and allows for an integrated instructional approach. The curriculum should allow students to develop life skills and apply content knowledge within a real world context. STEM education is active and focuses on a student-centered learning environment. Students engage in questioning, problem solving,collaboration, and hands-on activities while they address real life issues. In STEM education, teachers function as classroom facilitators. They guide students through the problem-solving process and plan projects that lead to mastery of content and STEM proficiency. STEM proficient students are able to answer complex questions, investigate global issues, and develop solutions for challenges and real world problems while applying the rigor of science, technology, engineering, and mathematics content in a seamless fashion. STEM proficient students are logical thinkers, effective communicators and are technologically,scientifically, and mathematically literate. (4)
>
> There are two goals for STEM education in *high school*. The first goal is on the development of STEM proficient students. All students will continue to grow in their STEM proficiency as they progress from grades 9-12. Students demonstrate independence and become more focused and sophisticated in their approach to answering complex questions, investigating global issues, and developing solutions for challenges and real world problems. STEM proficient students graduate with the basic skills and knowledge required to pursue post-secondary study or work in any field.
>
> The second goal for STEM education in high school is on the advanced preparation of students for post-secondary study and careers in science, technology, engineering, or mathematics. High school provides a unique opportunity for students to explore different career paths and college majors through advanced coursework, career academies, magnet programs, STEM academies, specialized STEM programs, internships, and dual enrollment opportunities. Specific programs to address the needs for advanced preparation of students shall be determine by individual schools systems. (5)

This curriculum seeks to merge the boundaries of science, technology engineering and math, while connecting these subjects to arts and the humanities. Each student will explore STEM through enriching and authentic hands-on learning opportunities.

Domains

Throughout the program, the student will achieve objectives in nine domains.

- **Algorithms**

  The student will design, analyze, and evaluate algorithms to solve authentic problems.

- **Programs**

  The student will use the commands, statements, procedures, and conventions of a text-based interpreted language [Python] to independently and collaboratively plan, compose, debug, run, edit, and document software that addresses an authentic purpose for a user or community with something at stake.

- **Data**

  The student will classify, store, retrieve, manipulate, query data sources. (revise to match CAS)

- **Computers**

  The student will define the components of a computer system and articulate its architecture.
  Correlary Standards, Benchmarks, Objectives:

  - CompTIA (A+, Strata, Linux+ Powered by LPI) learning objectives;
  - Cisco IT Essentials learning objectives;
  - McRel Benchmarks for Business Education (21 - 30)

- **Technology and Culture or Digital Literacy**

  The student will explore, interrogate, and hypothesize about causal relationships between technology and culture [to include public policy, values, economics]
  Correlary Standards, Benchmarks, Objectives:

  - MD (MSDE) Fundamentals of Technology Curriculum
  - Common Core > English Language Arts Standards > Science and Technical Subjects > Grades 9 & 10
  - Common Core > English Language Arts Standards > Science and Technical Subjects > Grades 11 & 12
  - Common Core > English Language Arts Standards > Reading Literature > Grades 9 & 10
  - Common Core > English Language Arts Standards > Reading Literature > Grades 11 & 12
  - New Media Literacies: A Syllabus (Henry Jenkins)
  - Bay, Jennifer. New Media (Purdue Syllabus). Web. 3 February 2013.

- **Electronic Engineering**

  The student will design, test, diagram, install, repair, and troubleshoot electronic systems and components.
  Correlary Standards, Benchmarks, Objectives:

  - McRel Benchmarks for Engineering Education (Standards 1 - 4)
  - MD (MSDE) Fundamentals of Technology Curriculum

- **Post-Secondary Transition Support**

  The student will explore and contrast post-secondary professional and academic opportunities.
  Correlary Standards, Benchmarks, Objectives:

  - Common Core College and Career Readiness Standards for Reading
  - Common Core College and Career Readiness Standards for Writing
  - Common Core College and Career Readiness Standards for Listening
  - Common Core College and Career Readiness Standards for Language

- **Networks**

  Understand ethernet and internet architecture and protocols; configure and administer network services for an authentic purpose. [placeholder]
  Correlary Standards, Benchmarks, Objectives:

  - CompTIA Network+ and Security+ Learning objectives
  - CCNA (Cisco) Learning Objectives

Primer Seminars

Professional Collaboration and Consultation

This transitional course prepares students for the academic assessments and professional content of the program. It is intended as a summer transition program to be taken at the recommmendation of an academic advisor or based on pre-assessments.

The student will:

- Produce professional writing with an authentic purpose and audience with something at stake [memos, emails, letters of interest]
- Interview a client to produce a needs inventory and shape client expectations
- Practice and demonstate fundamental professional skills [e.g. punctuality, attendance, articulation, eye contact, strategic awareness and use of personal space, stance, etc.]
- Use nonverbal communication to support a unified purpose
- Collaboratively write and edit authentic technical documents using a collaboration and revision system [e.g. helpdesk FAQ content]
- Speak articulately, confidently authoritatively, and concisely
- Evaluate sources for authoritativeness, reliability
- Collaboratively articulate an academic honesty and professional ethics policy
- Digital consultation: synchronous and asynchronous tools and conventionsy

Digital Literacy

This transitional course prepares students for the academic content of the program by introducing them to fundamental characteristics of both web-based communication and the consumption and efficient use of productivity applications. Digital literacy is to be taken at the recommendation of an academic advisor based on interviews, recommendations, and preassessments.

The student will:

- Demonstrate proficiency with the user interface of [GUI] browsers with significant currency.
- Navigate and search Web sites with a command line browser
- Identify a correctly formatte URL, navigate to an URL efficiently, and conduct a search from the browser interface effectively search lesson plans and objectives from Google]
- Place holder
- Command line text editor place holder
- IDE orientation placeholder
- Word Processor place holder
- beyond the web, ftp https, irc placeholder

1 Connected learning is an approach to addressing inequity in education in ways geared to a networked society. It seeks to leverage the potential of digital media to expand access to learning that is socially embedded, interest-driven, and oriented toward educational, economic, or political opportunity. Connected learning is realized when a young person is able to pursue a personal interest or passion with the support of friends and caring adults, and is in turn able to link this learning and interest to academic achievement, career success or civic engagement. This model is based on evidence that the most resilient, adaptive, and effective learning involves individual interest as well as social support to overcome adversity and provide recognition (Ito). The approach knits together three crucial contexts for learning: interest-powered; peer-supported; academically oriented. In addition, it embraces these key design principles: production-centered; open networks; shared purpose.

- Computer assisted diagramming and semantic mapping [placeholder; illustrative example is dia]
- Navigate the file system
- File nameing conventions
- Diagram filesystem hierarchy
- Differentiate between Save and Save As

Year 1

Artifacts of Learning (examples)

The student will:

- Assemble an ATX computer system according to authentic specifications;
- Access a computer remotely (command line and GUI);
- Create an electronic device based on a provide schematic;
- Administer a computer with a network operating system (GNU/Linux);
- Use the stages of the writing process to produce an assessment of the impact of an innovation on culture;
- Sequence steps in a process;
- Compose an analysis of multiple effects of a single cause;
- Compose an analysis of the causes of a single effect;
- Maintain a nightly journal of substantive responses to nonfiction & fiction reading in the content area;
- Create semantic maps to represent relationships between content;
- Diagram circuits according to convention;
- Accurately enter code in a shell scripting language;
- Visually and verbally articulate the common components of systems (including environment and noise);
- Given a shell script, annotate with effective and elaborative comments;
- Plan, compose, execute, debug a shell script that prompts for input and produces conditional output;
- Maintain a hardcopy log of changes to a system under her administration;
- Successfully emulate a computer system and use it to produce a manipulate a filesystem [resize partitions, write partition table].
- Plan, diagram [according to convention] and create a working electronic circuit using prototyping tools.
- Research and develop a students' guide to shell scripting language that compares and contrasts at least three shell languages.
- Provide credit to those whose ideas and content has been used in creating new works.
- Given a kit, build a volt-ohm-meter to be used to complete course projects.
- solve simple mathematical problems with inductive reasoning

Learning Objectives

Level 1

The student will:

- Differentiate between science and technology;
- Define *system* and components common to systems [including environment and feedback];
- identify and define the boundary of an common system;
- Define technology in terms of tools, ideas, methods, actions;

- read and narrate existing storyboards depicting familiar activities;
- sequence a collection of images conveying a familiar process;
- identify and catalog everyday devices that respond to signals and instructions;
- recite the design process [recursive];
- verbally and graphically articulate a definition of a computer system;
- visually identify the components of a computer system;
- identify and label input and output devices of a computer system;
- identify and differentiate input and outport ports on computer;
- connect the physical components of a computer system;
- Articulate the common functions of an operating system;
- Install an operating system based on documentation;
- Configure global system settings (e.g. keyboard config, locale, time-zone) of an OS;
- interact with the computer though both the command line and graphical user interfaces;
- Define and classify software [system and application software];
- Given a schematic and electronic components, create a simple electronic toy [series];
- Articulate a community problem or need that can be solved or satisified by computer system;
- Write a proposal for a computer system that can solve a problem are satisfy a need;
- analyze a technological innovation and evaluate its impact economic, political, cultural impact.
- Recognize the ethical and social implications of computer use. (College Board)
- Practice keyboarding;
- Demonstrate computer start-up and shut-down procedures;
- Demonstrate the execution of an existing program in a text-based language
- Explain the storage, retrieval, and deletion of programs;
- graphically and verbally differentiate between ohms, amps, volts;
- Measure ohms, amps, volts.
- Make sense of problems and persevere in solving them
- Articulate a definition of algorithm and algorithms' role in programming
- Articulate visually or verbally the foundations of algorithmic problem solving
- Explain motherboard components and features [CompTIA A+ Essentials]
- Classify power supply types and features [ComptTIA A+ Essentials]
- Explain the purpose and characteristics of CPUs and their features [CompTIA A+ Essentials]
- Distinguish between different display devices and their characteristics [CompTIA A+ Essentials]
- Install and configure peripherals and input devices [CompTIA A+ Essentials]
- Given a scenario, integrate common preventative maintenance techniques [CompTIA A+ Essentials]
- Evaluate and select appropriate components for a custom configuration, to meet customer specifications or needs [CompTIA A+ 220-801]
- Install an appropriate power supply based on a given scenario [CompTIA A+ 220-801]
- Evaluate and select apporpropriate components for a custom configuration, to meet authentic specification or needs [CompTIA A+ 220-801, modified]
- Identify connector types and associated cables [CompTIA A+ 220-801]
- Given a scenario, use appropriate safety procedures [CompTIA A+ 220-801]

- Explain environmental impacts and the purpose of environmental controls [CompTIA A+ 220-801]
- Map the ways the changing media landscape has impacted the way young people learn [Jenkins]
- Measure electricity and resistance
- Handle and connect electronic components without overloading, damaging, or destroying them [Make:Electronics]
- Read and understand a problem description, purpose, and goals [AP]
- Explain the use of virtual machine technology to run multiple operating systems concurrently [Cengage *Parallel* 1];
- Describe the hardware components of a personal computer system [Cengage *Parallel* 1];
- Describe the peripheral components of a personal computer system
- Contrast virtualization and emulation [Cengage *Parallel* 1];
- Describe the preventative maintenance for a computer system [Cengage *Parallel* 1];
- Enumerate common environmental threats to a computer system;
- Connect and test a personal computer system [Cengage *Parallel* 1].
- Describe the historical milestones of GNU/Linux and Microsoft Windows [adapted from Cengage *Parallel* 2];
- Describe the architecture of common PC operating systems [adapted from Cengage *Parallel* 2];
- Identify the functions of an operating system [adapted from Cengage *Parallel* 2];
- Describe the interaction between an operating system and its components [Carswell, *Parallel* 2]
- List popular productivity and system utility applications for GNU/Linux and Windows [adapted from Carswell *Parallel* 2];
- Customize GUI desktops for a Microsoft operating system and a GNU/Linux distribution [adapted from Cengage *Parallel* 3];
- Access data from a graphical user interface [adapted from Cengage *Parallel* 3];
- Launch an application from a graphical user interface [adapted from Cengage *Parallel* 3];
- Install and configure productivity applications and system utilities [adapted from Cengage *Parallel* 4];
- Uninstall software.

Level 2

The student will:

- collaborate to determine a "system administrator code of conduct";
- Independently represent the collaborative process (verbally or visually);
- escalate privileges with a network OS account;
- Control read, write, execute file and directory permissions from the command line [mode];
- manage files, directories, and removeable media [ls, mount, umount, rm, rmdir];
- Differentiate command line commands, parameters, and arguments;
- manage users and groups [usermod, useradd, adduser, etc.];
- customize the user environment;
- configure the computer for remote access to the command line and graphic user interface [ssh, xrdp, vnc];
- strategically determine and set a computer hostname;
- Articulate and contrast features and use scenarios of VNC and RDP servers;
- identify the computer's unique network address from the command line (ip address, dhcp);
- Remotely access the computer and account [with ssh and rdp];
- install software using a package manager;

- manage software installation, updates, and removal;
- search for software in a repository [apt-cache search];
- create storyboards depicting personal narratives and everyday activities;
- Use the design process to create and issue direct commands to make things happen with technology;
- identify simple problems that can be solved using programmable tools, toys, or systems;
- use the design process to solve simple problems with programmable tools, toys, or systems;
- Set up and configure networking services including DHCP and NTP;
- Configure localization settings to tailor the user environment to the locale;
- classify items in simple sets of data;
- use a Web browser to shop competitively for hardware and software components;
- Identify, requisition, build a hardware solution to determined specifications to solve an identified problem;
- Install and configure a software solution to solve identified problem or satisfy authentic need;
- articulate environmental threats to hardware and practice preventative care.
- Demonstrate keyboarding progress through increased speed and accuracy
- Electronics projects from kits
- Given values, determine volts, watts, ohms, amps using Ohm's law.
- Reason abstractly and quantitatively
- analyze and design simple algorithms
- Identify how participatory cultures work to support the growth and contributions of their members [Jenkins]
- Classify switches and relays [Make: Electronics]
- Define capacitance [Make: Electronics]
- Describe the characteristics of three Windows file systems [Carswell *Parallel* 5];
- Describe the characteristics of four GNU/Linux file systems
- Mount a file system in GNU/Linux and Windows [adapted from Carswell *Parallel* 5];
- Manage file systems in Windows 7 and GNU/Linux [adapted from Carswell *Parallel* 5];
- Describe directory structures [Carswell *Parallel* 6];
- Display directory structures [Carswell *Parallel* 6];
- Create, remove, and rename directories [adapted from Carswell *Parallel* 6];
- Use and recite file management commands;
- Use removeable drives for the storage of data [adapted from Carswell *Parallel* 6]
- Describe the contents of files and identify associated applications [adapted from Carswell *Parallel* 7];
- Display, interpret, and appy file attributes [adapted from Carswell *Parallel* 7];
- Fild files based on their names or content [adapted from Carswell *Parallel* 7];
- Understand the functions of common text editors [Carswell *Parallel* 8];
- Work with multiple files in text editors [Carswell *Parallel* 8];
- Demonstrate effective use of cut, copy, and paste commands [adapted from Carswell *Parallel* 8];
- Search for character strings in documents [Carswell *Parallel* 8];
- Search and replace character strings in documents [Carswell *Parallel* 8];
- Describe the features of command-line interpreters [Carswell *Parallel* 9];
- Use the command line to access help files for commands [adapted from Carswell *Parallel* 9];
- Display the contents of files [Carswell *Parallel* 9];

- Contrast available shell interpreters;
- Create scripts to automate simple tasks [Carswell *Parallel* 9];
- Manage tasks with task manager and process managers [adapted from Carswell *Parallel* 10];
- Monitor and evaluate performance [adapted from Carswell *Parallel* 11];
- Define and accurately use networking terminology [adapted from Carswell *Parallel* 12];
- Display and interpret TCP/IP settings [Carswell *Parallel* 12];
- Access network shares;
- Display and determine folder and file-sharing permissions.

Level 3

The student will:

- Identify and articulate similarities between storyboards of everyday activities;
- Use the design process to plan a linear (non-branching) sequence of instructions;
- develop and improve a sequence of instructions (write a shell script);
- Make a file executeable;
- given a set of data, present data in a systematic way;
- View, control, and kill processes, manage process priority, and load and unload kernel modules;
- install software from source;
- create and access a personal code repository using a revision tracking system;
- Read flowchart;
- Given diagramming software, create a flowchart for provided and self-produced program.
- Compose, revise, and debug a shell script using a command-line text editor;
- Strategically annotate a program written in a text-based language [functional or descriptive comments];
- Repurpose existing code in a text-based language and modify to solve a different, authentic problem than intended.
- Automate and schedule (shell scripts, at, cron) routine administrative tasks
- Demonstrate increased keyboarding speed and accuracy.
- use prototyping resources, including breadboards, to design purposeful circuits.
- Design, debug a text-based program to programmatically determine ohms, watts, volts, amps from given values.
- Articulate the way electricity is used to control lab equipment and computer systems.
- Construct viable arguments and critique the reasoning of others
- Use appropriate tools strategically
- Attend to precision
- Look for and make use of structure
- Look for and express regularity in repeated reasoning
- articulate asymptotic and standard notations as a growth of functions
- Recognize and be able to respond to core debates surrounding the value of bringing new media technologies and participatory culture practices into the classroom [Jenkins]
- Summarize a science-fiction cultural artifact.
- Transplant electronic components from a breadboard onto a perforated board [Make: Electronics]

Year 2

Artifacts of Learning (examples)

- Integrate a 1-wire environmental sensor into a computer system and log its data using an efficient and appropriate database tool. Use mathematical operators to programmatically convert celsius measurements to fahrenheit. Graph environmental sensor data over time using rrdtool, graphite, or cacti.
- Given a compatible scanner, hardware GPIO array and NOS, create a scan-to-email solution that works with a single button push. (obviate costly network scanner solution)
- Given a code base for Flesch-Kincaid readability, create an in house, cross-platform text document analyzer that assesses a writing samples grade levels based on at least three indicis/algorithms: use web/cgi solution or package for distribution using interpreted langauge.
- Develop and publish a unique Linux distribution that serves an authentic purpose or solves a real-world problem.

Learning Objectives

Level 4

The level four student will collaboratively rely on the design process to draft, test, debug, and deploy an original, short program in a text-based language that programmitcally solves an authentic problem for a user. The student will choose a license purposefully and develop appropriate clear technical documentation for a user with something at stake.

The student will:

- analyze and symbolically represent a sequence of events;
- research, write, and publish a student guide to software licensing;
- identify and classify different types of data (text; integer; decimal; instruction);
- understand the need for care and precision of syntax and typeography in given instructions;
- Enter instructions with demonstrated care and precision for syntax and typeography;
- give instructions involving selection and repetition;
- Interpret an algorithm and verbally predict an output;
- present data in a structured format suitable for processing;
- Design and implement solutions to problems by writing, running, and debugging computer programs (visual, compiled, interpreted) [College Board]
- Develop and select appropriate algorithms and data structures to solve problems (College Board)
- Illustrate a process using a flowchart conventionally
- Demonstrate the use of pseudocode
- Use a soldering iron to create designed circuits based on schematics
- Strategically choose from among a variety of "divide and conquer" methods to solve recurrences
- practice probabilistic analysis of randomized algorithms
- Understand the hardware and software components that make up networked computer systems and how they interact
- Explain how networks such as the Internet work
- Understand how computers can monitor and control physical systems
- Outline some of the ethical challenges which youth face in their roles as media producers and members of online communities [Jenkins]
- Narrate the development of computing technology from vaccuum tubes through mainframes to parallel and cloud computing;
- Articulate the relationships between science, technology, and science-fiction literature.

- Plan and develop a solution that relies on an API
- Cultivate and practice a daily news habit to keep informed of current events in science, technology, engineering, mathematics [bemedialiterate.com]
- Increase efficiency and simplify electronic circuits with ICs.
- Contrast high-level programming with low-level programming;
- Contrast compiled languages and interpreted languages;
- Use electronics test and measurement instruments to troubleshoot electronic devices;
- Read schematic electronics diagrams for purposes of testing and development

Level 5

The student will:

- decompose a problem into its sub-problems and make use of notation to represent it;
- analyze and present an algorithm for a given task;
- recognize the similarities between simple problems and the commonality in the algorithms used to solve them;
- explore and verbally or visually articulate the effects of changing the variables in a model or program;
- use the stages of the design process to develop, test, and refine sequences of instructions and show efficiency in framing these instructions;
- generate written verbal critiques of programs that will serve future projects;
- given a problem, manipulate strings to generate programmed verbal output;
- given a problem, select and use appropriate data types to program a solution;
- Characterize and use simple (1-dimensional) data structures.
- Code fluently in a text-based paradigm consisting of several classes and interaction objects. (College Board)
- Demonstrate familiarity with, and be able to use, standard library classes (College Board, modified)
- identify acceptable names for variables in a text-based language
- recognize and apply the symbols for mathematical operations in a text-based language
- demonstrate the use of for and while loops and
- demonstrate the use of conditionals
- Control electronic devices programmatically through GPIO and serial interfaces
- Use heapsort to sort and order statistics
- use quicksort to sort and order statistics
- sort data in linear time
- Develop programs that accomplish given goals, including controlling or simulating physical systems
- solve problems by decomposing them into smaller parts
- use sequence, selection, and repetition in programs
- work with variables and various forms of input and output;
- evaluate alternative algorithms and designs to solve a single problem
- detect and correct errors in algorithms and programs
- Describe our current understanding of the connections between participatory culture and civic engagement, including the relationship between the digital divide and the participation gap [Jenkins]
- Comprehend the framework of basic social skills and cultural skills associated with the new media literacies [Jenkins]
- Reflect and respond to science-fiction literature addressing themes of technological utopias/dystopias, sentience, cybernetics and cyborgs, or cyberspace.

- Develop, write, debug, execute a program written in a compiled language;
- Analyze passive electric circuits to predict their behavior;
- Analyze active electronic circuits, such as amplifiers, to predict their behavior;

Year 3

Artifacts of Learning (examples)

- Computer power control (electronic solution)
- Create custom Linux distribution that serves a specific purpose for a specific audience
- Collaborate on an open source project and earn credit as a team member by documenting, testing, decoding.
- Write a code philosophy for an original project
- Create a portfolio of stocks of five corporations integral to the technology landscape and available for public offering; graphically track the performance of the portfolio; publish opinion on performance of an assett in relation to current technology events
- Resume/CV/Portfolio; professional membership. Foster a professional identity and network using linkedin or other services.
- Earn CompTIA Linux+ Powered by LPI Certification

Learning Objectives

Level 6

The student will:

- analyze, comprehend, and predict the output of complex algorithms (e.g. sorting or searching algorithms);
- Use diagrams to describe systems and their components
- fully decompose a comple problem into its sub-problems and make use of a notation to represent it
- given simple problems, recognize similarities and a model which fits some aspects of the given problems.
- use programming interfaces to make predictions and vary the rules within the programs;
- Assess and articulate the validity of a self-produced program by considering or comparing alternative solutions;
- independently write a short program that solves a problem;
- independently debug a self-composed program;
- produce programes that rely on procedures with parameters and functions returning values;
- programmatically manipulate 1-dimensional arrays;
- use and interpret 2-dimensional data structures;
- use the design process to use 2-dimensional data structures to solve a problem.
- Survey, contrast, evaluate post-secondary program types within CS/Engineering/Software Studies/Media Studies, etc;
- Survey, contrast, evaluate technology certification routes, with consideration for experience, learning objectives, career readiness;
- Evaluate technology sector job opportunities and represent the relationships between skills and marketability verbally or visually
- read and understand a large program consisting of several classes and interacting objects; read and understand a description of the design and development process leading to such a program (College Board)
- Rely on the features of a programmed microcontroller solution to a problem
- Contrast elementary data structures

- create hash tables
- create a binary search tree
- recite the properties of a red/black tree (algorithms)
- augment data structures
- Design, use, and evaluate computational abstractions that model the state and behavior of real-world problems
- write structured programs using procedures
- use data structures such as tables or arrays
- explain how an algorithm works, and why it represents an efficient solution to the problem
- use at least two textual programming languages
- identify and investigate forms and examples of new media as well as the theories that underlie and emerge from these forms [Bay]
- Analyze science-fiction literature in terms of social and political issues and stakes.
- Articulate and contextualize historical and contemporary contributions of nondominant cultures to the STEM fields;
- Use and implement commonly used data structures.
- Demonstrate an understanding of digital circuits;

Level 7

The student will:

- verbally and visually describe key algorithms and evaluate them for efficiency (e.g. sorting/searching parity);
- fully decompose a problem into its sub-problems and make error-free use of an appropriate notation to represent it;
- recognize and articulate similarities in complex problems and produce a model which fits some aspects of the problems;
- Rely on the design process to build a system that relies on pre-constructed models of bode;
- design and use complex data structures (including relational databases);
- select, compare, and use programming tools suited to their work in a variety of contexts;
- translate specifications expressed in natural language into the form required by a programming tool;
- Articulate and contrast the benefits and limitations of programming tools and of the results they produce;
- Given a problem, independently program a maintainable solution in a text-based language;
- Independently debug a self-produced program written in a text-based language;
- Analyze, simplify, and use complex data structures in self-produced programs.
- Prepare a resume/CV with well-suited career objective and cover letter for a specific audience and purpose.
- identify the elements of dynamic programming
- Compose a social and political critique of a science fiction artifact.
- Narrate the history and articulate the structure and functions of mass media
- Develop and select appropriate algorithms and data structures to solve problems;
- Explain the physical principals involved in electo-mechanical energy conversion and describe the contruction of electrical motors and generators;

Level 8

The student will:

- independently select appropriate programming constructs for specific tasks, taking into account ease of use and suitability to audience and context;

- articulate and represent similarities in complex problems and produce a model that fits most aspects of these problems;
- independently write a program for others to use;
- apply advanced debugging procedures to self-produced programs;
- use a documentation system to document usage and features of a self-produced program for others;
- normalize data structures;
- analyze and verbally represent the relationship between complex real life and the algorithm, logic and visualisations associated with programming.
- identify the elements of the greedy algorithms
- Work creatively on individual and collaborative projects in a range of digital systems
- create, reuse, revise and repurpose digital information and content with attention to design, intellectual property, and audience.
- Explain how instructions are executed within a computer system
- explain how data of many types can be represented and manipulated in the form of binary digits
- reflect on the personal, social, economic, and ethical impacts of technology and technological change, and the implications for rights, responsibilities, and freedoms
- Demonstrate familiarity with New Media theories and the process of mass communication
- Apply digital media literacy strategies to use media to empower herself in [digital] participatory communities. [bemedialiterate.com]

Year 4

All participating students have the opportunity to take qualifications [2] in aspects of information technology and computer science which lead to progressio high levels of study or a professional career. Whether or not they take up this opportunity, participants will achieve the learning objectives below. (*Draft ICT Programme of Study*)

Artifacts of Learning (examples)

- Collaborate on a successful open source project: provide support, documentation, debugging, etc
- In an interpreted language, write a cross-platform application that stores and retrieves data...user interface...original application, license, documentation. Published solution.

Learning Objectives

Level 9

The student will:

- independently recognise and varbally articulate similarities between complex problems and produce a general model that fits aspects of them all;
- competently and confidently use a general-purpose text-based programming language to produce efficient solutions to problems;
- Collaborate with a community of programmers to produce, debug, document a software product with a specific purpose
- Provide support for a software product
- Independently incorporate bug tracking system into workflow
- Independently maintain a well-documented and clearly licensed code repository using a revision system
- Identify characteristics of advanced data structures (b-trees, fibanachi heaps, etc)
- Create elementary graph algorithms

- Deploy course concepts in the development of an independent research project which makes a substantive scholarly contribution [Jenkins]
- Summarize and critique core theorists working in the field of New Media Literacy [Jenkins]
- Propose/submit a professional conference paper, workshop, or presentation
- Demonstrate digital media literacy skills to access, analyze, evaluate, and create digital content and social action strategies to engage and empower citizens in a diverse and participatory digital democracy.

Authentic Learning & Assessment

Note

Authentic assessment is a foundational value of this curriculum (framework?). Each community will have different problems to solve, so the specific assessments of learning and instruction will have to be determined by determined by the instructor, preferably in consultation with the students. Below are some examples of authentic assessments we have either worked-through together or have queued.

Research-Based Pedagogy

Resources and supporting research will be listed here. There's a collection of resources at http://badsville.ignorelist.com.

Examples

- Create a LAN segment
- Create a WAP to allow wireless access to the lab-network segment.
- Create a router to isolate and manage lab traffic
- Create a reset circuit (momentary switch) for Raspberry Pi B r 2.
- Create a climate sensor system with 1-wire and Raspberry Pi technology for server cabinet
- Collect, log, graph data from 1-wire network, publish to web (rrdtool, graphite, cacti, nginx or apache2); administer server solution
- Create LCD or LED IP address display for headless Raspberry Pi [Adafruit or LXF]
- Built an ATX computer system and configure as virtualization platform
- Install and configure epoptes-client on lab machines
- Install and configure xrdp on lab machines running Linux
- Create user accounts for students; grant admin privs appropriately
- Build a scan to email solution using Raspberry Pi and sane-compatible scanner
- Install Adafruit WebIDE on Raspberry Pi
- Install and configure IDE of choice on lab workstations (NetBeans or Eclipse)
- Install vim-gnome on Ubuntu and vim on Raspberry Pi for vim-tutor
- Install and configure in-house gitlab
- Create custom Ubuntu LTS distro with specific suite of applications with accessibility in mind; productivity applications to include dia, calligra, pdfedit, etc.
- Create and publish a custom Raspbian appliance

2 “Qualifications” means that participating students cannot merely be offered a token course leading to nothing of value.

Anchor Texts

Text-Based Computer Languages

- *Hello, World* (Python, Manning)
- *Ruby for Absolute Beginners*
- *Computer Programming for Teens* (C+)
- *Head-First JavaScript* (O'Reilly Media)
- *Learning JavaScript* (O'Reilly Media)
- *Head-First Python* (O'Reilly Media)
- *Head First PHP & MySQL* (O'Reilly Media)
- *Python for Kids* (No Starch Press)

Linux and Shell Scripting

- *LPI Linux Certification in a Nutshell* (O'Reilly Media)
- *CompTIA Linux+ Certification Powered by LPI* (Axzo Press)
- *Shell Scripting* (Wrox)
- *Command Line* (http://en.flossmanuals.net/command-line/)
- *Guide to Parallel Operating Systems with Windows 7 and Linux*

Visual Languages

- *Super Scratch Adventures* (No Starch Press)
- *Scratch 1.5* (Badger)

Hardware and Electronic Engineering

- *Make: Electronics* (Platt)
- *The Art of Electronics* (Hill and Horowitz)
- *Student Manual for the Art of Electronics* (Hayes and Horowitz)
- *Understanding Computers*
- *A+ Guide to Hardware* (Andrews)
- *A+, Network+, Security+ Exams in a Nutshell* (O'Reilly Media)
- *Technology and Engineering*

Raspberry Pi

[Software and Hardware Engineering]

- *Raspberry Pi Education Manual* (CAS)
- *Raspberry Pi Users' Guide* (Upton)
- *Programming the Raspberry Pi* (Simon Monk)
- *Getting Started with the Raspberry Pi* (Richardson)

Post-Secondary Transition

- *Programming Python* (O'Reilly) [placeholder]
- *Expert Resumes for Computer and Web Jobs, 3rd Ed* [placeholder]
- *Hacking the IT Cube* [placeholder]

Miscellaneous

- *Pragmatic Guide to Git* (Pragmatic Bookshelf)
- *Introduction to Algorithms 3rd Ed* (Cormen et al)
- *The Design and Analysis of Algorithms*
- *Algorithms Unplugged*
- *Digital Media Revisited* (MIT Press)

Prospective Curriculum Technologies

- 1-wire
- rrdtool
- Gitlab, Github, Bitbucket
- Agile Development
- Bugzilla
- docuwiki
- mediawiki
- python-sphynx
- reStructuredText
- Markdown
- SCRUM Development
- GNU/Linux
- Subversion
- Git
- bzr (bazaar)
- Trac
- Virtualization Platform
- Observium
- Graphite
- Cacti
- Apache2
- nginx
- chromium-browser
- vim, emacs
- netbeans
- eclipse
- AdaFruit WebIDE
- Python

- Ruby
- PHP
- Scratch
- JavaScript
- Windows 7
- SSH
- RDP client [remmina]
- PuTty
- Filezilla
- Variable Termperature soldering station
- Dia
- Calligra Flow
- bash
- Idle
- drpython
- scite
- gedit
- SublimeText 2
- virtual machines (VirtualBox, VMWare, Xen)
- qemu
- epoptes-client, epoptes
- etherpad-lite with syntax-highlighting plugin
- SQLite, MySQL, PostgreSQL
- IRC (irssi, irc.freenode.net)
- trac
- html

References

Computing at School Working Group. Computer Science: A Curriculum for Schools. (2012)

Ito, Mizuko, et. al. *Connected Learning: An Agenda For Research and Design*.

Ito, Mizuko, et. al. *Connected Learning: An Agenda For Research and Design: Summary*.

Popham, W. James. "Performance Assessment." *Classroom Assessment: What Teachers Need to Know*. 6th ed.

MSDE. *Maryland State STEM Standards of Practice Framework Instructional Guide Grades 9-12* (Draft)

Draft ICT Programme of Study. November, 2012.

Computer Science Curriculum 2008: An Interim Revision of CS 2001. Association for Computing and Machinery IEEE Computer Society.

Jenkins, Henry. New Media Literacies: A Syllabus . Web. 3 February 2013.

Bay, Jennifer. *New Media* (Purdue Syllabus). Web. 3 February 2013.

College Board. *Computer Science A Course Description*

Indices and tables

- *genindex*
- *modindex*
- *search*