heading

version 0.0.1

preface

January 23, 2013

Contents

Curriculum Proposal

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]

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 Computer Science, Electrical Engineering, and Information Technology.

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)

Assumptions

Computer Science and Electrical 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)

It is assumed that students are exposed to productivity applications and assistive technologies across the curriculum. This program builds on that most fundamental IT experience to help students develop into critical consumers and producers of digital tools.

Domains

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

- **Algorithms**

  The student will define the term algorithm and identify authentic uses for algorithms.
- **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**

  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
- **Electronic Engineering**

  The student will create, test, diagram, and troubleshoot electronic circuits 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
- 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
-

First Year

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.

Learning Objectives

Level 1

The student will:

- Define *system* and components common to systems [including environment and feedback];

- identify and define the boundary of an common system;
- 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.

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

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.

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 and linux, create a scan-to-email solution that works with a single push button. (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.

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 (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

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

Year 3

Artifacts of Learning (examples)

- Placeholder

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)

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.

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.

Year 4

Artifacts of Learning (example set)

- Place holder

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

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 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

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)

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*
- *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)
- *Understanding Technology*

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]

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*.

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

Indices and tables

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