

Proposal to Create Hamline Plan Requirement – “D”

Creative Problem-Solving with Digital Tools

January, 2013

Part One – Background and Proposal

Our taskforce was convened to evaluate the current Computer-Intensive (c) Hamline Plan requirement.

During our conversations with faculty inside and outside of this taskforce we heard two consistent questions:

- 1) How can we ensure Hamline graduates have exceptional technological skills that are relevant to their chosen field of study, to support their success in the workplace and in future academic work?
- 2) What kind of technology-oriented knowledge, if any, is critical to the education of a liberal arts student of today?

As stated in our C Revision Proposal, our taskforce believes that the current C insufficiently addresses these questions as a) it does not support extensive technology-oriented skill development within a major/program, requiring only one course that is typically at the 1000 level and often outside of major and b) it is defined so broadly that it fails to tie the C to a liberal arts education in a meaningful and coherent way. In response to the first of these concerns, we proposed the elimination of the current C requirement. This change, however, does not address the question of technological skills more broadly relevant to the liberal arts student of today.

Our taskforce explored the potential intersection of technology/computer knowledge and liberal arts education. Our recommendation is to create a new general education requirement “D” representing “Creative Problem Solving with Digital Tools” with the following description:

Given that computing devices play a large and increasing role in how we participate in the world professionally and socially, a knowledge and understanding of how these devices are made to function is both liberating and empowering. Each Hamline student will take at least one course in which computing technology is used in creative and problem solving contexts. Courses satisfying this requirement would ask the student to identify a problem that can be solved, at least in part, through the development of a technological solution. Students will be asked to design, develop and test this solution, creating a custom approach that leverages the power of computing technology in a larger context. These courses are likely to be found in areas such as digital media, natural sciences, STEM-oriented education, business analytics, computer science or any other discipline that can benefit from custom technology to support exploration and problem solving. When appropriate, these courses would also discuss issues of ethics, security, and intellectual property issues as they relate to digital creation.

The new D would not be fulfilled by simple use of software packages, proprietary or otherwise. The student would create a custom solution (using macros, coding, HTML, or other tools) to “look under the hood”, causing the computer to perform new tasks dictated by the student in the context of academic coursework. Examples of existing courses that could already fulfill the requirement include quantitative analysis in business, Stephen Kellert’s first year seminar course on interactive media, digital arts courses, calculus, David Hudson’s topics in professional writing course as well as introduction to programming (please see page 2 of this proposal for illustrative course descriptions).

As this is a proposal for the development of a new letter over the next few years, there would be time for faculty development for any interested faculty who seek to include this letter in a new or existing course. Faculty who would like to use text-mining in history or literature, explore robotics, or build custom survey and data analysis tools for a nonprofit organization could all align with the new letter. Faculty desiring such a component in their courses would be supported in its development by Hamline’s Quantitative Reasoning Center as well as the Center for Teaching and Learning. Like most other Hamline Plan requirements, the decision to offer a course of this type is entirely up to the discretion of the faculty and departments/programs are not required to have a course in this curricular area.

Learning outcomes – a student taking a D course will be able to:

- Design a logical plan for the development of a custom technological solution
- Implement design of a custom technological solution, diagnose design and implementation problems, and generate appropriate solutions
- Communicate effectively with technical and non-technical audiences
- Discuss issues of ethics, security, and intellectual property, as appropriate to digital creation

The new D, combined with an increased emphasis on program-specific technology skills, will allow Hamline graduates to truly be competent and responsible in their use of technology, using that technology to solve problems in ways that are innovative, analytical and ethical. Furthermore, this bold new general education requirement will support Hamline's promise to create students who "engage independently and reflectively in lifelong learning", as the act of creation in a digital realm will generate confidence and knowledge to allow for active continued participation in this realm beyond graduation.

As indicated, there are already courses offered at Hamline that align with the proposed D learning outcomes. What follows are course descriptions for three existing courses that illustrate possibilities within this curricular area.

Topics in Professional Writing: Writing and Design for the Web and New Media – Prof. David Hudson

ENG 3370: Topics in Professional Writing: Writing and Design for the Web and New Media was piloted as a computer-intensive blended online course in Fall 2012. It focuses on written and visual communication that meets the changing needs and expectations of new media users. Students who take the course, many of whom are adept *consumers* of the web and other new media, are introduced to the computer coding that allows them to become *producers* and *creators* of new media. For example, consumers can add content to web pages or even create their own web pages with programs that write HTML and CSS code for them, but these limit the range of expression to those prescribed by the application's designer. Perhaps more importantly, they also obscure the underlying structure and functioning of the web. On the other hand, requiring students to understand the way the web works and to learn the languages in which it is written allows them to not only "peer behind the curtain," as the proposed Hamline Plan revision suggests, but has them actively engaged in "producing the show."

Operating at the producer level involves a multi-stage process as envisioned in the proposed learning outcomes. Along with mastering the technical skills, students learn how to apply their knowledge in a project that requires them to research potential user-scenarios, develop a plan that solves an organizational problem, create a website design, produce all component materials, and then upload their work online and present it to the class for critical evaluation. A key aspect of this is that students understand that the acquisition of specific tools, in this case the latest versions of HTML and CSS, is not the ultimate goal. Both languages will be obsolete in a few years. The point is to "learn how to learn" in a technical context, allowing them to acquire new knowledge as it emerges and, in turn, teach it to others. In these ways the computer requirement can help "create students who 'engage independently and reflectively in lifelong learning.'"

Interdisciplinary Perspectives on Interactive Entertainment – Prof. Stephen Kellert

In our First Year Seminar on "Interdisciplinary Perspectives on Interactive Entertainment" students completed a two-week unit on the design and programming of videogames. This involved our teaching apprentice, a computational science major, presenting a brief introduction to the practice of coding and the C++ programming language. Each student completed two short programming exercises in order to demonstrate their familiarity with the most basic elements of writing a program.

Then we divided the class into three groups: gameplay, narrative, and coding. The class as a whole was tasked with designing a text-based adventure game, and the students had to work within their groups as well as communicate between groups in order to develop a workable game that was fun, playable, and

simple enough to code in a short time. The exercise gave them hands-on experience with the challenges of designing, building, and revising a piece of computer software.

Web Design – Prof. Dave Ryan

DMA 1460: The first objective of the course is to achieve a sound understanding of the crucial concepts of web design including: structure and processes of web site publishing, basic HTML tags, use of WYSIWYG web editor (Adobe CS6 Dreamweaver), rudiments of visual design for the web, and rudiments of interactive design for the web. The second objective is to apply that knowledge through the production of a professional web site. Students will spend the term planning, designing, building, posting and testing their web sites.

This course gives students the basic technical skills and conceptual framework for creating engaging web sites using HTML 5 and CSS 3. The course covers an overview of internet operation, maximizing design for an interactive web environment, optimizing media for web use, site management. Through an examination and discussion of the best of current web design, students will sharpen perceptual and critical thinking skills. Students will spend the term planning, designing, building and revising a professional site.

Part Two – Supporting the Revision Proposal

This proposal allows us to reinvigorate our teaching and learning around the general principle that students need specific technological skills for professional purposes *and* creative technological knowledge that allows them to thrive as citizens, consumers, labor force participants, and people. Re-envisioning a general education computer/technology requirement in terms of *Creative Problem Solving with Digital Tools* encourages students to be adaptive and participatory in the digital world, rather than passive consumers. When the Hamline Plan first included a “Computer Intensive” requirement, it was considered ground-breaking to expect all students to achieve facility with the use of computers. Now, what our students need is the ability to use computers as more than a “black box” that makes things happen when they click on a button; they need to know how to make the computer do something that it is not already set up to do.

Gaining this skill will require students to peer behind the curtain and overcome the sense that information technology involves a mystery that only an elite can access. In this way, it will fulfill one of the oldest and noblest goals of the liberal arts: giving students the ability to operate as citizens unconstrained by pre-existing structures. Each student will be able to say to a potential employer or graduate program that they designed and created a technological “product”, following the project from design through completion. This will make them stand out from other applicants.

Not surprisingly, we are not the first to ask what computer/technological knowledge is appropriate within a liberal arts education. Much of the discussion in this arena has focused on the value of student experiences with computer programming. While our proposal is not limited to programming, the anticipated skills and knowledge parallel those that would be gained from exposure to coding. Donald Knuth made a case to insert computer programming into the liberal arts in his classic article “Computer Programming As An Art” in 1974. Knuth argued that programming deserves its proper place alongside arts of old “because it applies accumulated knowledge to the world, because it requires skill and ingenuity, and especially because it produces objects of beauty.”

The New York Times featured Princeton professor Brian Kernighan in 2002 in an article entitled “To the Liberal Arts, He Adds Computer Science”. Kernighan’s course, *Computers in our World*, is taught to liberal arts undergraduates and requires them to make a simple web page and write some simple computer programs. The end goal, according to Kernighan, is to demystify technology and give students the tools to “think more intelligently about this technology for themselves”. Beyond these technical skills, Kernighan introduces topics of privacy, copyright and antitrust issues that are increasingly relevant to knowledge creation.

The Chronicle of Higher Education has recently published articles on the intersection of computing knowledge and liberal arts education. In April of this year, Robert Talbert contributed articles commenting on Georgia Tech’s programming requirement for all undergraduate students. Talbert contends that a collection of courses with varied levels of difficulty (courses in Office applications for non-techy students and sophisticated programming courses for students entering with

higher skill levels) violates the purpose of higher education institutions by creating tiered educational experiences based on point of entry. Talbert also argues it underserves students as Office application knowledge is “necessary but no longer sufficient”. Rather, developing a suite of courses that each embed programming knowledge in variety of contexts (as Georgia Tech has done) meets student needs/interests and represents good curricular design.

In his article “Programming vs. ‘Technology’”, Talbert further presses for programming as a general education requirement, arguing that computer-oriented general education requirements have weakened over time and now provide students with trivial experiences. In contrast, he describes programming as “the intellectual pursuit of agency over the computer and a means for solving problems”. A recent book by Douglas Rushkoff, *Program or Be Programmed*, touches on this same theme – agency over the computer in this increasingly digital world. Nathan Gilmour, in his review of this book, cites a central theme of the book - true education should teach people to do things that other people haven’t thought of before. As applied to technology and computers, this means we must go beyond teaching students to operate and use existing technology created by others. According to Gilmour:

“Rushkoff insists that there’s no place in a democratic society in the computer age for one class of programmers and a much larger class of end-users; like literacy and mathematics, to acquire a working knowledge of computer code is simply to know the fabric of the civilization that citizens are supposed to help run.”

Prompted by NYC mayor Bloomberg’s announcement that he plans to learn to code, Chronicle blogger Eric Charles discussed the reasons why coding is important to students of all stripes. Charles suggests multiple reasons for programming’s place in the liberal arts, among them that coding:

- “requires a particular attention to detail that you cannot weasel your way out of”,
- “teaches a new way of thinking”, similar to the acquisition of foreign language skills, and
- “builds a toleration for failure”.

On this last point, Charles is echoing a previous Chronicle article by Professor Paula Krebs entitled “Next Time, Fail Better”. Krebs argues that most of her humanities students lack the healthy experience with failure and frustration that is embedded in programming. She notes that “each time their efforts fail, the developers learn something they can use to get closer to success the next time”. Reinforcing this point, Charles notes that “writing bad code is how you start writing good code”, and coding is therefore uniquely able to teach this critical combination of exploration and creativity with persistence and attention to detail.

Our taskforce believes that all students should apply critical thinking skills to the realm of technology. While programming is one way to do this, digital creation more broadly embraces this same skill set. As computers and technology change rapidly, the new “D” will engage students in an experience that will apply across time, as the same process will be used in the present and the future to design technological solutions, using the toolsets available.

Part Three – Implementing the Revision Proposal and Assessing Learning Outcomes

For the new Hamline D, development of new courses or adjustments to existing courses could take some time, hence the suggested development of this letter over the next few years. Interested faculty could be identified, and professional development opportunities offered to implement the D inside existing or new courses. Ideas for course units could be piloted in first year seminars. The Center for Teaching and Learning and/or the Quantitative Reasoning Center can support faculty in their own professional development as well as in course development. Courses will need to be carefully assessed to assure that the intent of the new D is fulfilled and this will need to be done by a committee of experts.

A potential rubric for the proposed D is provided in Appendix A. The course professor will be responsible for assuring that some form of assessment consistent with the rubric will take place.

Charles, E. (2012, May 16). Is computer programming an important part of a liberal arts education? *fixingpsychology.blogspot.com*.

Gilmour, N. (2011, August 24). Programming as a Liberal Art: A Review of Program or Be Programmed by Douglas Rushkoff. *christianhumanist.org*.

Knuth, D. E. (1974, December). Computer Programming as an Art. *Communications of the ACM, Volume 17 Issue 12*, pp. 667-673.

Krebs, P. M. (2012, May 6). Next Time, Fail Better. *The Chronicle of Higher Education*.

Lohr, S. (2002, October 31). To the Liberal Arts, He Adds Computer Science. *The New York Times*.

Talbert, R. (2012, April 6). How Not to Require Computer Science for All Students. *The Chronicle of Higher Education*.

Talbert, R. (2012, April 9). Programming vs. "Technology". *The Chronicle of Higher Education*.