The Final Project

When you learn about programming, you don’t get very far by reading books or listening to lectures. You have to sit down at the computer and write a few programs on your own. Recognizing the intellectual dimension of computer science takes a similar investment of time and energy. You have to wrestle with the ideas. For this reason, the focus of this Sophomore College course is less on the material I present in class than it is on the work that you yourselves will do in the context of the class project.

The project deliverables are as follows:

• *A 25-minute oral presentation to the class.* The goal of the presentation is to give your classmates an overview of your topic area, illustrate the topic with some simple examples, and offer a sense of why that topic is intellectually exciting. You are free to use the classroom projector to illustrate your presentation—presumably including parts of the web site described below—but it is important to avoid becoming so wrapped up in the technology that the main points are lost. As shown on the syllabus, the presentations will be on Tuesday, September 15. It is important to make sure that you don’t run over the 25-minute limit so that there can be time for a couple of questions and to set up for the next presentation.

• *A web site that gives a more detailed presentation of the topic.* Your web site should include a top-level general introduction along with links to more detailed information, some of which will be part of your site and some of which will be resources elsewhere on the web. We will assign groups so as to distribute the existing web-authoring expertise as evenly as possible.

There are also the following milestones:

**Thursday, September 3**  
*Topics due.* Most groups in the past have chosen one of the topics described in this handout. You may, however, come up with a topic of your own, but you’ll need to convince me that your topic offers the same level of challenge and embodies a similar level of intellectual excitement.

**Thursday, September 10**  
*Annotated bibliographies due.* The first step toward researching your topic is to look up resources on the topic from a variety of sources: books, journals, the web, and faculty members in the department. Make a guess at the resources you’ll be using, write it down, and mail it to me before the end of Thursday.
Project teams

I put together a set of project teams that balances as closely as possible your web proficiency as you rated yourselves yesterday.

<table>
<thead>
<tr>
<th>Team 1</th>
<th>Farzaana Adam, Jose Antonio de los Heros, Caroline Debs, Zhaolin Ren</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team 2</td>
<td>Paul Crews, Amy Liu, Christina Ramsey, Edward Daniel</td>
</tr>
<tr>
<td>Team 3</td>
<td>Michael Arruza-Cruz, Harry Elliott, Juliana O’Donohue, Katherine Stowell</td>
</tr>
<tr>
<td>Team 4</td>
<td>Kevin Balchoo, Jose Saldana, Tristan Vanech</td>
</tr>
</tbody>
</table>

Suggested topics:

1. *The invention of computers: Who gets the credit?* For most of the second half of the 20th century, credit for creating the first electronic computer in the United States was generally assigned to J. Presper Eckert and John Mauchly, who developed the ENIAC computer at the Moore School of the University of Pennsylvania in 1945. For their work, Eckert and Mauchly applied for and received a patent on the ENIAC design. The question of who actually invented these concepts, however, remains a subject of controversy. From 1937 to 1942, a research team at Ohio State University consisting of John Atanasoff and Clifford Berry developed a computer—usually called the ABC or Atanasoff-Berry Computer—that had many modern features, including the use of binary arithmetic. John Mauchly corresponded with Atanasoff beginning in 1940 and, in 1941, spent several days at the Iowa State campus learning about the machine.

In 1973, after a protracted legal battle, a federal judge invalidated the ENIAC patent and gave credit to Atanasoff for the invention of the digital computer. The issue, however, remains controversial, with many arguments as to how much the Eckert-Mauchly team actually took from the earlier design.

A project on this topic should include the following topics:

- A discussion of the structure and operation of the ABC and the ENIAC, at a level that allows the audience at the presentation to understand how they work.
- An analysis of the similarities and differences between the two systems.
- A synopsis of the legal history.
- A discussion of other areas in which credit is disputed for early progress in computation. In this discussion, you might, for example, consider whether the stored-programming concept is indeed original to John von Neumann and the fact that the designers of computer hardware, such as Eckert and Mauchly, received widespread attention while the people who programmed those systems, almost exclusively women, were largely ignored.

2. *Proofs of program correctness.* In many areas in computing, practice has been improved substantially by exploiting theoretical results. One area in which the success of this marriage of theory and practice has been more mixed is in the field of program semantics, which seeks to capture the meaning of a program in mathematical terms and then to prove that the program implements its formal specification. In a fascinating 1979 article entitled “Social Processes and Proofs of Theorems and
Programs,” U.S. computer scientists Richard DeMillo, Richard Lipton, and Alan J. Perlis argued that the idea of proving programs correct is fundamentally misguided, generating a firestorm of controversy among computer scientists on both sides of the Atlantic. That paper and its myriad responses in the letters pages of Communications of the ACM over the following year (all of which was reprised a decade later after a follow-on article by James Fetzer appeared in the same journal) define what I think is one of the most interesting philosophical debates in the history of computing.

3. Cellular automata. Some of you may be familiar with Life—a game invented by British mathematician John Horton Conway and popularized in Martin Gardner’s “Mathematical Games” column in Scientific American. Conway’s life game is played on an unbounded two-dimensional grid of squares, each of which is either empty or marked with a counter, corresponding to the states dead and alive. The game evolves over time in a series of discrete generations in which the fate of each cell depends on the cells in its immediate neighborhood, which consists of the eight surrounding squares. According to the standard rules, a cell is alive in the next generation if either (1) it was alive in the current generation and has two or three live neighbors or (2) it was dead in the current generation and has exactly three live neighbors. For example, a grid consisting of three live cells in a line—which is called a blinker—oscillates back and forth from generation to generation as shown in the following diagram:

```
  O O O    O O    O O O
```

The cells at each end of the line die because they have only one live neighbor; the cell in the center survives because it has two. The cells to the side of that center square have three live neighbors and thus are born in the next generation.

An even more interesting pattern is the glider, which moves diagonally across the screen in a cycle of four patterns:

```
  O O O    O O    O O O
```

Because many patterns in Life have extremely fascinating evolutionary histories, it is great fun to play with as a solitaire game, seeing what interesting patterns you can produce and how they evolve. The game, however, is also interesting as a model of computation. A project report on the Life game should include the following:

- A review of some of the standard Life patterns, such as blocks, beehives, blinkers, and gliders.
- An overview of the early history of the Life game, when many questions remained unresolved about its nature. For example, one of the critical early questions was whether any patterns would grow without limit. Bill Gosper answered this question in the affirmative by producing a pattern he called a glider gun, which
fires off a succession of gliders sailing off to infinity.

- A discussion of questions that remain open about Life patterns.
- An explanation of why cellular automata are as powerful as Turing machines as a model of computation.
- A discussion of generalizations of life to one- and three-dimensional spaces.
- Examples that show how cellular automata—presumably with different rules—can be useful in practical applications.

4. **Excursions in a two-dimensional world.** In 1884, Edwin A. Abbott published *Flatland: A Romance in Many Dimensions*—a fantasy about higher dimensional spaces (along with a fair bit of social satire) that has never yet gone out of print. Over the years, other authors have taken up the same thing, as in *Sphereland: A Continuing Speculation on an Expanding Universe* written by Dionys Burger in 1965 and the more recent *Flatterland: Like Flatland Only More So* published in 2001 by Ian Stewart. While these books are primarily concerned with mathematics, there is one adaptation—*The Planiverse: Computer Contact with a Two-Dimensional World* written by A. K. Dewdney in 1984—that has particular relevance for computer scientists. In *The Planiverse*, Dewdney describes how it is possible to build logical circuits and computer hardware in a two-dimensional world.

A report on this topic should include at least the following:

- An overview of the story.
- A discussion of engineering issues in the Planiverse and an explanation of how things work there, particularly including the logic circuits.
- At least one original engineering design for the Planiverse beyond what Dewdney describes.

5. **The history of object-oriented programming.** Many of the fundamental ideas of object-oriented programming were developed by the Norwegian team of Ole-Johan Dahl and Kristen Nygaard in the 1960s and are reflected in the design of SIMULA-67 and, to a lesser extent, the even earlier SIMULA I language. What did those ideas look like in the beginning and how have they evolved? Why did it take so long for those ideas to catch on in the mainstream? To what extent are these ideas still controversial?

6. **The open-source movement.** Over the last two decades, there has been growing enthusiasm for the open-source movement, which seeks to make software freely available rather than commercial. Individual success stories, such as Linux, Mozilla, and Apache, have intensified this trend. A project on this topic might cover the following issues:

- The philosophy of the open-source movement and an analysis of how that philosophy has evolved since the publication of *The GNU Manifesto* in 1985.
- The scale of the open-source movement today.
• An analysis of the underlying economics, focusing in particular on why open-source appears to have been successful at all, when essentially all economic analyses predicted its failure.

• An assessment of its impact on the software industry.

• Some investigation of the sociology of the open-source movement and the nature of the people it attracts. It was sobering, for example, to read a paper at the Grace Hopper Celebration of Women in Computing several years ago that suggested that less than one percent of the participants in the open-source community are female—a percentage far lower than the already scandalously low percentage in the industry.

If none of these topics excites your group, feel free to think of something else. Here are a few ideas, which clearly need to be fleshed out a bit:

• The evolution of numerical algorithms, both before and after the rise of computing
• Symbolic mathematical manipulation in systems like Mathematica
• Strategies for managing concurrent execution
• Intellectual ideas behind hacking and defenses against it
• Design and development of special-purpose graphics hardware