Pomona College
Department of Computer Science

A Guide through the Senior Exercise

Everett L. Bull, Jr. and David Kauchak

April 18, 2019

Submitted as part of the senior exercise for the degree of
Bachelor of Arts in Computer Science

Professors Kim Bruce and Tzu-Yi Chen, advisors

Abstract

Completion of a senior exercise in one's major is a graduation requirement at Pomona College. In Computer Science, this involves a senior seminar course and i) when following the old upper-division requirements, attending all research colloquia throughout the senior year, ii) when following the new upper-division requirements, attending 12 research colloquia in both junior and senior years. Some students may also optionally choose to participate in one of three additional senior experiences: a project, a thesis, or clinic at Harvey Mudd. This document serves as a guide to assist students in planning and executing their senior exercises.

Acknowledgments

The author is deeply grateful to Professors Bruce, Chen, Kauchak, and Sood for their patient support and generous suggestions, and to Patrick McNally for careful copy-editing.

Contents

List of Figures

List of Tables

Preface

The Computer Science Department welcomes you to the senior exercise! We want you to have a rewarding and memorable experience as you complete your major with a valuable, rigorous, and fun final endeavor.

The *Pomona College Catalog* [Pom11] states, "Each student's major will culminate in a senior exercise designed to deepen understanding and integrate the content and method of his or her field of study." Besides being a vehicle that deepens understanding and integrates content and method, the senior exercise ought to be a labor of love. We encourage you to begin thinking about what you want to get out of your senior year. Speak to other students and faculty members and investigate all the possibilities.

The senior exercise in Computer Science involves the senior seminar course (CS190) as well as attending CS colloquium throughout your senior year, if following the old upper-division requirements or attending 12 talks during both junior and senior years when following the new upper-division requirements. There are also three *optional* experiences that some of you may choose to undertake that extend upon your coursework and the senior seminar: a project, a research thesis, or clinic at Harvey Mudd. Chapter 1 gives some more details about all of these options and latter chapters of this guide provide more specifics.

In the spring of your junior year, you will enroll in the Senior Seminar course. Additionally, you should start thinking if you would like to do any of these three optional exercise listed above. Some of these require you to make decisions at particular dates (e.g. if you want to do clinic, you must register for clinic in the fall).

In this guide, we have concentrated on the *decisions* you must make and the *mechanics* of the process, and we have tried to avoid being overly prescriptive about the *nature* of an individual undertaking. We hope that this guide will be helpful as you plan, execute, and complete your senior exercise. With thoughtful (and early) planning, your senior exercise will proceed smoothly. Please read this guide carefully and refer to it often;

its pages contain experience and advice collected by students and faculty members over the past years. If you do find yourself in a crisis, refer to the suggestions in Appendix B.

Chapter 1

Premeditation

There are some shared requirements that all Computer Science seniors must undertake to graduate. Those are listed below in section 1.1. Some students may also optionally decide to undertake one of the three senior exercise extensions. To help you decide if any of these options would be interesting to you, most of this chapter is devoted to detailed outlines of these three options, including the requirements, the schedule, the advisor, and the grading policies. There are some redundancies among the descriptions because we want each one to be self-contained.

Read these outlines in conjunction with the list of approximate dates in Appendix A. The specific dates for the current academic year will be announced in the Senior Seminar, over e-mail, and on the Department's web site.

1.1 Common Features

Each senior must take Computer Science 190, *Senior Seminar,* which is a separate graduation requirement. This course involves reading, discussing and presenting research papers as well as writing a expository research paper.

Each senior that follows the old upper-division requirements must attend all the departmental colloquia, which are held approximately every other week on Thursday at 4:15 pm. Seniors following the new requirements, should attend 12 talks in their junior and 12 talks in their senior year. Talk with the faculty member who is overseeing the senior exercise for missed colloquia.

1.2 Optional Exercises

In addition to the senior seminar and colloquium, some students may also optionally choose to do one of three additional senior exercises. If you choose any of these routes, you will have an advisor who is a member of the Pomona College Computer Science faculty. The advisor for the senior exercise may or may not be the same as the student's academic advisor. The role of the advisor will vary according to the option, but in all cases, the advisor will guide the student through the "local requirements," including the progress meetings, the presentation, and the written work.

Each of the three options will require you to sign up for additional courses in your senior year. See below for details on which those are for each option and if they will count for credit towards the major.

1.3 Project

A project is carried out (normally) in the spring semester of the senior year, with some of the preparation being done in the preceding semester. A project can consist of software development, library research, or theoretical study. In all cases, there will be significant background research.

The role of a project advisor varies according to the student, the advisor, and the topic. The advisor is sometimes quite directive, often a source of valuable advice, and frequently a sounding board for the student's ideas. In any case, you should expect to meet with your advisor regularly, at least weekly, over the course of the project. You and your advisor should develop a detailed schedule to keep the project on track.

The advisor must be a faculty member in the Computer Science Department of Pomona College. You may, of course, seek advice from faculty members in other departments or at other colleges. In some cases, students have worked most closely with someone other than their official departmental advisor.

Matching a student with an advisor and a topic is a complicated process that depends, in part, on the student's interest. The student should, normally, have taken one or more advanced courses in the general area of the topic, and the advisor should have some expertise in that area. The student and the advisor should be comfortable working with one another. Also, because we want to divide the workload evenly among advisors, each potential advisor has a limit on the number of advisees. In mid-September students are asked to submit a ranked list of three or more advisor-project

pairings, and with understanding and good will, we expect to come to a fair and beneficial assignment. In any case, each student that decides to do a senior project will have an advisor and a topic by about the third week of the fall semester of the senior year.

Requirements for the Project

The requirements for the project option are as follows:

- Computer Science 190, *Senior Seminar,* in the fall semester. In addition to the other work of the seminar, each student will
  - ◇ submit a list of advisor-topic pairs in the first weeks of the semester;
  - ◇ complete the background research for the project and submit an annotated bibliography and a literature review at some point in the second half of the semester;
  - ◇ acquire and install any software, data, or tools that are needed for the project; and
  - ◇ submit an extended abstract with a specific description of the proposed project's purpose and methods.
- Participation in the (approximately monthly) progress report meetings, both semesters. In the fall semester, the meetings will be part of the Senior Seminar.
- If following the old upper-division requirements: Attendance of all the (approximately biweekly) departmental colloquia, both semesters.
- If following the new upper-division requirements: Attendance of 12 (approximately biweekly) departmental colloquia, both during junior and senior years.
- Computer Science 192, *Senior Project,* in the spring semester. Most of the work on the project takes place in the first two-thirds of the semester. Each student will
  - ◇ submit, shortly after the middle of the semester, a substantial draft that includes all the sections of the paper;
  - ◇ give a formal presentation, near the end of the spring semester, of the project, experience, and conclusions; and
  - ◇ submit a paper which reports on the project and its results.

This document itself serves as a model for the form of the project report; see Chapter 4.

Evaluation of the Project

Computer Science 190 and 192 count as one-half course each.

A student's grade in Computer Science 192 is determined jointly by the Computer Science faculty members who consider six components of the project:

- The goal of the undertaking. Was it appropriately ambitious?
- Preparation. Did the literature review and other preparatory activities cover the appropriate prior work in the area?
- Execution. Were the efforts (both in quantity and quality) sufficient? Were all the intermediate and final deadlines met?
- Evaluation of results. Was the process of evaluation appropriate and thorough?
- Presentations. Were the oral and written presentations clear, understandable, and complete? Keep in mind that presentations are graded on *both* content and delivery.
- Participation. Did the attendance and contribution to group meetings, conferences with the advisor, and the final presentation meet faculty expectations?

These items are interrelated, and there is no set number of "points" assigned to each one. In particular, there is an interaction between the first and third items. A wonderful execution of a trivial goal may not be worth more than a weak outcome on a more challenging goal. And a ridiculously challenging goal may preclude you from any success at all. The key is to find the right balance.

Further, you should keep in mind that a successful project need not be one that is clearly better than all previous work in the area. We often learn as much or more from experiments that fail as from those that succeed. On the other hand, we do expect you to make your project as successful as possible within the time constraints.

To state the criteria in another way, an **A-range** project is interesting, well-conceived, ambitious, and successful. There is a significant result, although it need not exactly match the original goal of the project. Any

obstacles that arise are handled thoughtfully and creatively. All work is submitted on time. The presentation and the paper are clear, polished, and well-organized. In order to receive an A, all the components—conception, preparation, execution, presentation, and written work—must be of A quality.

A **B-range** project is fairly ambitious but only partly successful, or else it is less ambitious and solidly successful. There are results, but they may be incomplete or tentative. All the requirements and deadlines are met. The presentation and the paper or thesis are clear and organized.

A **C-range** project is insubstantial or falls short of its goals; there are few definite results. All the requirements are met, even if some of the work is a little late. The presentation and the paper or thesis are disorganized or occasionally muddled.

A **failing** project is poorly conceived and inadequately executed. It is lacking substance that might lead to a solid and interesting conclusion. Some of the work may be seriously late or missing altogether. The presentation and paper are ungrammatical or difficult to follow.

In addition to the grades in the various courses, the Pomona College Senior Exercise is graded "not passed," "passed," or "passed with distinction." The designation appears on the transcript. It is rare, but not without precedent, for a student to fail the senior exercise and therefore not graduate. At the other end of the spectrum, "distinction" is reserved for a small number of really outstanding senior exercises. A grade of A in the senior exercise course does not guarantee a rating of "distinction" on the senior exercise.

1.4 Group Project

Starting in 2010, the Computer Science department started offering the option of a group project allowing students to work on a joint project with up to five students. The procedural details for the group project are similar to those for the project.

Requirements for the Group Project

The requirements for the group project option are as follows:

- Computer Science 190, *Senior Seminar,* in the fall semester. In addition to the other work of the seminar, the members of the group will

- ◇ learn the fundamentals and design principles that will guide the project,
- ◇ become competent with the software tools to be used,
- ◇ create a detailed design of the system to be created, and
- ◇ assign specific spring-semester tasks to team members.

- Attendance at regular meetings with the group and its advisor. The team will meet weekly in the fall semester and more frequently in the spring. The advisor will attend at least one meeting a week.
- Participation in the (approximately monthly) progress report meetings, both semesters. In the fall semester, the meetings will be part of the Senior Seminar.
- If following the old upper-division requirements: Attendance of all the (approximately biweekly) departmental colloquia, both semesters.
- If following the new upper-division requirements: Attendance of 12 (approximately biweekly) departmental colloquia, both during junior and senior years.
- Computer Science 192, *Senior Project,* in the spring semester. Most of the work on the group project will occur in the first two-thirds of the semester. As with the regular project, there will be a paper that reports on the project and its results and a formal presentation. The extent and content of the paper and presentation will be determined in consultation with the advisor.

Evaluation of the Group Project

As in the regular project, Computer Science 190 and 192 count as one-half course each. A student's grade in Computer Science 192 is determined jointly by the Computer Science faculty members based on criteria similar to those for the regular project; see Section 1.3. The grade will be based both on the individual team member's performance *and* the overall effectiveness of the group.

1.5 Research Thesis

The research thesis option is for those students who want to investigate a topic more deeply. The topic is an aspect of the thesis advisor's research

program, and the work is carried out in close cooperation with the advisor. The finished product will be a substantial written thesis. A research thesis is a substantial undertaking that involves a full course in each semester of the senior year.

Students wishing to satisfy the senior exercise through the research thesis option must obtain permission from the department early in the spring semester of their junior year. Students who are interested in the thesis should approach a potential advisor in plenty of time to discuss the possibilities. Often preparatory research is done in the junior year or during the summer before the senior year.

The research thesis option is excellent preparation for a student who is considering graduate school. It is intended for students who are passionate about a particular topic, can work independently, and are well-equipped to pursue the project. Because of the intense and personal nature of the research thesis, the student and thesis advisor must mutually agree to work together. We cannot guarantee that every student will have the opportunity to do a research thesis.

Requirements for the Research Thesis

- Computer Science 190, *Senior Seminar,* in the fall semester. In addition to the other work of the seminar, each student will complete the background research for the thesis and submit an annotated bibliography, a literature review, and an extended abstract in the second half of the semester.

- Participation in the (approximately monthly) progress report meetings, both semesters. In the fall semester, the meetings will be part of the Senior Seminar.

- If following the old upper-division requirements: Attendance of all the (approximately biweekly) departmental colloquia, both semesters.

- If following the new upper-division requirements: Attendance of 12 (approximately biweekly) departmental colloquia, both during junior and senior years.

- Computer Science 191, *Senior Research and Thesis in Computer Science,* in the fall semester.

- Computer Science 191, *Senior Research and Thesis in Computer Science,* in the spring semester. Each student will

- ◇ submit, shortly after the middle of the semester (or earlier if the advisor requests it), a substantial draft that includes all the sections of the thesis;
- ◇ give a formal presentation, near the end of the spring semester, of the thesis, experience, and conclusions; and
- ◇ submit the completed thesis at the end of the semester.

This document itself serves as a model for the form of the thesis; see Chapter 4.

Evaluation of the Research Thesis

Computer Science 190 counts as one-half course. Computer Science 191 counts as a full course each semester. The fall semester of that course may count as an elective toward the major requirements.

The grading standards for Computer Science 191 are the same as for the project; see the descriptions on page 4. Obviously, the research thesis is expected to be longer and more substantial than the paper for a project. The thesis advisor will have a significant role in the determination of the grades for Computer Science 191.

As stated on page 5, students will also receive a grade for the overall senior exercise—not passed, passed, or passed with distinction.

1.6 Clinic

The Harvey Mudd College Computer Science Clinic is a program in which a corporation or research organization sponsors an academic-year-long effort to solve a real-world problem. A team of three to five students, together with a faculty advisor and a liaison from the client, works on the problem and provides written and oral reports to the client. For more information, see `http://www.cs.hmc.edu/clinic`.

A Pomona College student may satisfy the senior exercise requirement by participating in a clinic project. The clinic experience requires participation in both semesters of a Computer Science Clinic project at Harvey Mudd College. Students taking this option will work closely with the project advisor and the other team members. Each student will have, in addition, a Pomona College advisor to provide guidance through the local requirements.

The clinic option is well-suited for students who want to participate in a team effort or students who are interested in "trying out" a career in industry.

Computer Science 121, *Software Development,* is a prerequisite for the clinic.

Students wishing to satisfy the senior exercise through the clinic option must obtain permission from the department early in the spring semester of their junior year. An early commitment is necessary because Harvey Mudd College plans its clinic projects well in advance. Students must preregister in the spring to take Computer Science 183 in the fall semester of their senior year.

Requirements for the Clinic

- Computer Science 190, *Senior Seminar,* in the fall semester.
- Participation in the (approximately monthly) progress report meetings, both semesters. In the fall semester, the meetings will be part of the Senior Seminar.
- If following the old upper-division requirements: Attendance of all the (approximately biweekly) departmental colloquia, both semesters.
- If following the new upper-division requirements: Attendance of 12 (approximately biweekly) departmental colloquia, both during junior and senior years.
- Computer Science 183 and 184, *Computer Science Clinic I and II,* in the fall and spring semesters, respectively. A student must participate in all aspects of the clinic, including the reports and presentations to sponsors. The clinic is a full-year commitment; a student who enrolls in the fall is expected to participate through the completion of the project in the spring.
- An individual presentation, near the end of the spring semester, describing the clinic project, its successes and failures, and its results.
- A copy of the final clinic report. The clinic report will serve as the written record of the senior exercise; a separate individual report is not required.

Evaluation of the Clinic

Computer Science 190 counts as one-half course. Computer Science 183 and 184 each count as a full course. Computer Science 183 may count as an elective toward the major requirements.

Students following the clinic option are, of course, graded by their clinic advisor in Computer Science 183 and 184 in accordance with the policies of the Harvey Mudd College Computer Science Clinic.

As stated on page 5, students will also receive a grade for the overall senior exercise—not passed, passed, or passed with distinction. The grade will be determined by the Pomona College Computer Science faculty in consultation with the advisor of the clinic project.

Chapter 2

Preparation

This chapter is written *primarily* for students following the project and research thesis options. By the beginning of the fall semester of the senior year, everyone will have an advisor and a topic. The research thesis students will have made those decisions much earlier. As part of the work for the Senior Seminar, project and research thesis students will carry out background research and submit annotated bibliographies and literature reviews, as described in Sections 2.2 and 2.3.

Those participating in a clinic have already made decisions—one way or another—about advisors and topic, and the course of their work follows a different schedule. Everyone, though, can benefit from the comments on scheduling in Section 2.5.

2.1 Advisor and Topic

The first task is to find an advisor and a topic. Sometimes, students have a clearly-formulated topic in mind and look for an advisor who will help them refine and develop it. In other cases, students select an advisor first and then investigate various topics. There is nothing wrong with "shopping around" at the beginning of the process. Be willing to speak with several possible advisors and hear their suggestions for topics.

The choice of a topic requires careful thought. Keep an open mind as you search. Although many students write long programs, it is not necessary to write any code at all. A project or thesis might be a collection of data on how humans use computers, a study of legal or social issues connected with information technology, a statistical analysis of network performance, a proof of correctness or careful computation of complexity for an algorithm,

a design of an aspect of a programming language, or an exposition of one of the deep ideas in Computer Science.

Look for a topic that is interesting, deep, specific, and feasible. The topic should, first and foremost, be *interesting* to you. After all, you will be spending a whole semester (or a whole academic year, in the case of a research thesis) working on it. In the spirit of the Pomona College Senior Exercise, the undertaking should "deepen your understanding" and "integrate the methods and content" of Computer Science.

The topic should also be *deep* in the sense that it builds on the courses you have taken for your Computer Science major. A student who just finished a couple of introductory courses would not be prepared to complete a project or thesis on such a topic. The most successful undertakings are ones in which the student has completed one or more advanced courses in the area. It is also desirable to choose a topic close to your advisor's areas of expertise.

Generally, a more modest undertaking that is well-done is preferable to an ambitious but haphazardly-executed one. On the other hand, an utterly trivial task will not receive an A, no matter how well it is done.

The project or thesis proposal must be *specific* in that it attempts to answer a question, illustrate a principle, explain an important idea, or conduct an experiment. It is not enough just "to write a program" or "to try something and see what happens." When it comes time to present your work, you want to have a clear and specific conclusion.

As you evaluate possible topics, think about what kind of conclusion is appropriate. Is it a statistical analysis, a proof, a graphical display, or something else? Whatever it is, it will take some effort to create.

Finally, the project or thesis must be *feasible.* On one hand, you will be working on it for *a whole semester* (or *a whole year,* in the case of a thesis). On the other, you have *only one semester* to do it, and you will have other courses and activities during that time. And as you get to the end, you will have *only a few weeks* to finish it. Many projects and theses, but not all of them, involve intensive coding. If you are thinking of writing code, keep in mind that (1) coding takes longer than you think it will, and (2) there are other parts of the project that are equally important. Even if you do not plan to write programs, make realistic estimates of the time required to complete the project.

We all understand that your ideas and your work will evolve. Stay flexible and expect to change your direction slightly as you learn more about the subject. Your plans may even include decisions to be made on the basis of early work. Careful planning at the beginning will, however, reduce the

possibility that you will have to throw out everything and start over.

Your advisor can help you to achieve the right equilibrium among the competing qualities of interest, depth, precision, and feasibility.

2.2 Annotated Bibliography

Once you have decided on a topic, your first task is to begin collecting information about it. (That process may, in fact, begin earlier—as you are casting about for a subject.) It is highly unlikely that you are the first person to be interested in the subject. Find out what others have done. Who was the first to raise the general question? What are the significant developments in the subject? Have others tried to solve similar problems? If so, in what ways were they successful? What is the origin of your specific approach, and what are the competing approaches? How does your project differ from previous work?

Work with your advisor to get started. Often, once you find a few current articles, you can use their reference lists to work your way back in time to the original works on the subject.

For each article you find, keep a record of the full bibliographic citations and a summary of the article—anywhere from a short phrase to a few paragraphs. Figure 2.1 shows three annotated records: a journal article, a piece of software, and a submission to a conference.

Evaluate each article critically. Is there a well-defined problem or question? What is the article's conceptual or theoretical framework? Is the approach or methodology appropriate and effective? Are the experiments repeatable? (You might even try repeating one or two.) Is the presentation clear and logical? Do the data or the arguments support the conclusions? What contribution does the article make to your topic?

The annotated bibliography will, ultimately, contain all the works connected to your topic. You may omit those papers you found to be irrelevant. By the time you have completed the bibliography, you should have a sense that you really understand the development of the field and the current state of the art.

2.2.1 Quality of Sources

As you carry out your research, look for authoritative, reliable sources. They are the works that are released by reputable publishers, are written by authors with respected credentials, or have gone through rigorous peer-review processes.

[GJS76] N. E. Gibbs, W. G. Poole Jr., and P. K. Stockmeyer. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM J. Num. Anal.*, 13(2):236–250, April 1976.

Looks at improving the RCM algorithm [LiuSh76]. Basically original cm and rcm algorithms suggest trying *many* start nodes and taking the one which leads to ordering of smallest bandwidth. They show, in part, how to reduce the number of bfs's that are done.

Pseudo-diameter: run the bfs once, then take the nodes in the last level and run bfs again using each of those last nodes as the new root. Goal is to end up with two endpoints of a pseudo-diameter. They provide an algorithm for merging bfs trees from each endpoint (helps minimize width).

They demonstrate the algorithm on an assortment of structured grids.

[IPM] IPM: integrated performance monitoring. `http://www.nersc.gov/projects/ipm`.

Describes a tool that generates a detailed performance profile for parallel codes. Low overhead, no source code modification, summarizes computation and communication costs. [checked web site 6/1/07]

[SSl05] J. Schwartz, A. Steger, and A. Weißl. Fast algorithms for weighted bipartite matching. In S. E. Nikoletseas, editor, *Proceedings of the 2005 Workshop on Experimental Algorithmics (WEA 2005)*, volume 3503 of *LNCS*, pages 476–487, 2005.

Asked by a company to come up with a faster way to compute minimum weight perfect matchings on complete bipartite graphs ($n > 10000$). Standard algorithms ($O(n^3)$ for real weights, $O(\sqrt{n}m\log(nC))$ for integer weights) too expensive.

They experiment with discarding a large subset of the edges in the following way: first keep the smallest $(c\log n)$ smallest edges adjacent to each vertex, then add in $c'n\log n$ uniformly at random chosen edges of E. The second means that, by a theorem of Erdös and Rényi, the new graph has a perfect matching with high probability.

Then can use sparse algorithms to compute perfect matching. For arbitrary edge weights can get upper bound on achieved approximation factor based on choice of c. For uniformly distributed edge weights can show will get optimal matching with high probability based on a theorem by Frieze and Sorkin which bounds the heaviest edge in a min-weight perfect matching for a complete bipartite graph in this case.

In tests on real data, find if highly structured, need $c > 120$, otherwise 15 ok. c' was set small ($= 4$). If generate random uniformly distributed edge weights, cutoff value of $c = 4$ was enough.

Figure 2.1: Three entries from an actual annotated bibliography, courtesy of Professor Tzu-Yi Chen.

The following types of works are examples of unassailable sources:

- scholarly books;
- articles in well-known, refereed journals; and
- contributions to well-regarded, refereed conferences.

Your advisor can help you understand which journals are "well-known" and which conferences are "well-regarded." Conference papers are frequently followed by journal articles; in those cases, you should cite the journal article.

As a general rule, avoid citing a source that only appears on the web [Bul11]. The reference [AKS02] is an example of such a paper. It is, by far, preferable to cite the later publication [AKS04]. If a published work appears also on the web, it is helpful to the reader to give the url as well, as was done in [AKS04].

A valuable resource for your work is the CiteSeer database of articles in computer science. See `http://citeseer.ist.psu.edu` or the mirrors at `http://citeseer.csail.mit.edu` and `http://citeseer.ittc.ku.edu`. The Association for Computing Machinery's Digital Library, at `http://portal.acm.org/dl.cfm`, is available on campus (through financial support from Honnold Library), as are many of the publications of the Institute of Electrical and Electronics Engineers; see `http://www.ieee.org/web/publications/home/index.html`.

Some works do not appear on the web at all. This is particularly true for some of the seminal, ground-breaking papers from decades ago. As it is important to read and to cite these works, you will probably have to use some of Honnold Library's databases (just plain Google will not do the job!), and you may even have to set foot inside a library.

An exception to the "no web" rule is a work that is brand new and has not yet been published. For example, [AKS04] would not have been published in 2002, and *at that time* a citation of [AKS02] would have been acceptable.

Another possible exception to the "no web" rule is software. You may use software that you acquired from the web, and you should cite the source. The item labeled [IPM] in Figure 2.1 is an example of such a citation. There is often a published paper that accompanies the software (and is listed on the site with the software), and when it exists, you should include a reference to the paper is well.

The following works are potentially acceptable, depending upon the situation:

- technical reports from major universities,
- white papers from corporations,
- articles from newspapers and the popular press, and
- blogs or on-line essays from well-known authorities.

Technical reports are often preliminary versions of conference papers or journal articles. Cite the later, refereed versions when they are available.

Sometimes industry white papers are the only sources of authoritative information. Examples include the Java specifications from Sun Microsystems and the detailed Pentium specifications from Intel.

News articles and blogs can be especially helpful in providing a contextual setting or give a sense of the field at a particular moment. However, they are unlikely to be central to the main topic.

The following types of works are *never* authoritative:

- writings that appear only on the web and offer no evidence of reliability, and
- Wikipedia articles.

Remember that a web search is as likely to turn up the term paper of a high school student or the blog of a wacko as it is to find an article of a reputable expert from a known publisher. The reason that Wikipedia articles are never deemed reliable is that they are written anonymously by authors whose credentials are unknown; there is no reason to be confident that a Wikipedia article is complete or correct. Wikipedia may be useful, however, in leading you to more authoritative sources.

2.2.2 Creating the Bibliography

Use LaTeX and BibTeX to prepare your annotated bibliography. General information about LaTeX appears in Sections 4.3 and 4.4, and the BibTeX format is described in Section 4.5.11. The book [Lam94] contains a comprehensive introduction to BibTeX. The BibTeX source for the annotated bibliography of Figure 2.1 appears in Figure 2.2.

This may be your first experience with LaTeX. The biggest part of the task will be creating the bibliography file. The sample file `srpaper-sample-biblio.bib` provides an example of such a file, with many different kinds of entries.

```
@Article{GibbsPoSt76,
  author =  {N.~E.~Gibbs and W.~G.~Poole~Jr. and P.~K.~Stockmeyer},
  title =   {An algorithm for reducing the bandwidth and profile
             of a sparse matrix},
  journal = {{SIAM} J. Num. Anal.},
  year = 1976,
  volume = {13},
  number = {2},
  pages = {236-250},
  month = {April},
  annotate = "Looks at improving the RCM algorithm [LiuSh76].
    Basically original cm and rcm algorithms suggest trying
    \emph{many} start nodes and taking the one which leads to
    ordering of smallest bandwidth.  They show, in part, how
    to reduce the number of bfs's that are done.\par

    ..."
}

@Misc{IPM05,
  title = {{IPM}: integrated performance monitoring},
  howpublished = {\url{http://www.nersc.gov/projects/ipm}},
  annotate = {
    Describes a tool that generates a detailed performance
    profile for parallel codes.  Low overhead, no source code
    modification, summarizes computation and communication
    costs.  [checked web site 6/1/07]}
}

@InProceedings{SchwartzStWe05,
  author = {J.~Schwartz and A.~Steger and A.~Wei\ss{}l},
  title = {Fast algorithms for weighted bipartite matching},
  booktitle = {Proceedings of the 2005 Workshop on Experimental
               Algorithmics (WEA 2005)},
  year = {2005},
  pages = {476--487},
  editor = {S.~E.~Nikoletseas},
  volume = {3503},
  series = {LNCS},
  annotate = "Asked by a company to come up with a faster way
    to compute minimum weight perfect matchings on complete
    bipartite graphs ($n>10000$).  Standard algorithms
    ($O(n^3)$ for real weights, $O(\sqrt{n}m\log(nC))$ for
    integer weights) too expensive.\par

    ..."
}
```

Figure 2.2: The sources for the bibliographic items in Figure 2.1. The annotations have been truncated to save space.

Once you have the bibliography file, creating the final document is easy. Copy and rename the template file `annbib-template.tex` and then edit it to fill in a few fields. The comments will direct you to the relevant places in the template file. If you renamed the sample file to `mybib.tex`, then the command-line arguments to produce the document appear below; see Section 4.4. The result is the file `mybib.pdf`.

```
latex mybib
bibtex mybib
latex mybib
latex mybib
```

Please send copies of the completed bibliography to your advisor and to the instructor in charge of the Senior Seminar. See Appendix A for an approximate due date.

2.3 Literature Review

At the end of the fall semester, you will submit a literature review. It will normally consist of two to six pages of prose and a complete bibliography. If your annotated bibliography is already a complete compilation of the relevant sources, then all that remains to do is to write a clear description of the history of the subject—with citations. In many cases, you will identify one or two primary resources that you will consult in depth during the spring semester. The completed review will give you a clear sense of the context into which your work fits.

The literature review is a *draft* of the "background" chapter of your project paper. As you complete the project in the spring semester, you will no doubt run across other sources and include them in your summary of previous work.

The ability to carry out background research is important. When you have completed your bibliography and literature review, you will have developed valuable skills in finding resources and evaluating them efficiently and effectively.

Prepare the literature review using LaTeX and the `srpaper` document class with the `short` option. See Sections 4.3, 4.4, and 4.5 for instructions.

The literature review is due a few weeks before the semester ends. An approximate date appears in Appendix A; the exact date will be announced and posted. There will be cases in which the Computer Science faculty requests clarification or elaboration of the literature review; there is a second

date in Appendix A for the revision. As with the annotated bibliography, send copies of the literature review to your advisor and the instructor in charge of the Senior Seminar.

2.4 Extended Abstract

Along with the literature review, you will submit an extended abstract for your project. By the time you have completed the background reading and thinking, you should have a very concrete idea of your project.

The extended abstract might also be called a "research proposal." It is the sort of document that normally must be approved by supervisors (either in academia or industry) before a research project is undertaken, although in our case it is significantly shorter. It will be a separate document, about two or three paragraphs long, which describes the specific goals and strategies for the project. It should tell the reader

- what question you intend to answer,
- what you plan to do over the course of the project, and
- how you plan to draw conclusions from your work.

You may plan to write code, conduct an experiment, prove a theorem, design a system or a language, or study the development of an idea. Whatever you plan to do, be sure to include a strategy (including, usually, the collection of data) that will lead to a solid conclusion. Be sure to describe techniques for statistical analysis of your data if it is appropriate for your subject, as it will be, for example, if you are measuring the performance of a program or computer system.

Once approved, you should think of this as a contract between you and your advisor. Although it may be revised over time, it will guide the evaluation of your project. Even though it is only a page long, you should think carefully about what goes into it and discuss it thoroughly with your advisor.

Like the literature review, the extended abstract is to be prepared with the `srpaper` document class using the `short` option. See Sections 4.3, 4.4, and 4.5 for instructions.

The extended abstract is due at the same time as the literature review. The approximate due dates for the abstract and, if necessary, the revision appear in Appendix A. Send copies of the extended abstract to your advisor and the instructor in charge of the Senior Seminar.

2.5 Scheduling

Naturally, the largest part of your effort will go into the execution of the project or thesis. Because the projects are so different from one another, there is little we can offer in the way of general direction. You and your advisor will decide the course of the project. In the spring semester, you will work closely, and meet regularly, with your advisor to set intermediate goals, review progress, and solve any problems that arise. You will also attend group meetings, approximately monthly, to share news about progress, difficulties, and successes with your fellow students.

It is vitally important that you maintain a realistic schedule. Begin work right away, and try to make real progress in the first month. With your advisor, set intermediate goals and review the schedule frequently. As you work, some tasks will take more time than you expect and others (a few) may take less. Build into your schedule some slack for the "slippages" that always occur, and allow enough time for preparing the presentation and writing the paper.

Throughout your work, remember Hofstadter's Law [Hof79]. We are grateful to Jim Marshall for directing us to this fundamental truth of the universe.

> **Hofstadter's Law:** It always takes longer than you expect, even when you take into account Hofstadter's Law.

Chapter 3

Presentation

The project presentations occur late in the spring semester. Each student will give a presentation on one of the dates listed in Appendix A; the exact times and places will be announced. Each student will have 30 minutes: 20 minutes for the presentation and 10 minutes for questions.

3.1 Planning and Rehearsal

A good presentation requires careful planning and judgment. Your intended audience is the group of Computer Science majors from the junior and senior classes, but remember that those people have not spent the last few months intensely involved with your project. Think about how to introduce your work: what to include and what to omit.

In 20 minutes, you cannot say everything. It is acceptable, often highly desirable, to concentrate on just one aspect of your project. Be sure that you provide a clear statement of the problem and, later in the talk, a clear conclusion. Give enough important details so that the audience has an appreciation of your work but not so many that they become numb.

At the end of the talk, audience members should appreciate the central ideas and lines of reasoning in your project. Ideally, they will feel that they could, if they wanted, fill in the details for themselves.

Once you have an outline, it is time to begin rehearsing. Practice speaking out loud, while you are standing up, preferably with someone listening. You will discover that the presentation takes longer when you rehearse it in semi-realistic circumstances than when you whisper it to yourself. You will also discover the points for which you do not (yet) have good explanations. You are likely to revise your outline radically after one or two rehearsals.

After you have established the outline, it is time to refine the actual presentation. Work to overcome any bad speaking habits. Avoid speaking too softly or too quickly. Speak clearly and project to the rear of the room. Maintain eye contact with various members of the audience. Develop a relaxed but authoritative conversational tone. Again, several rehearsals—while you are standing and speaking aloud to another person—are essential.

You may want to rehearse before a video camera and evaluate your performance. If so, your advisor can arrange to borrow video equipment from ITS.

3.2 Visual Aids

Most students will use some kind of visual aid. Slides of the Powerpoint variety are most common, but other creative kinds of demonstrations are welcome.

Keep in mind that at least half of your audience's attention will be on the visual image and not on what you are saying. Choose slides or other demonstrations to supplement your words. If you have a chart, illustration, or animation that requires some attention, allow the audience time to absorb it. In the case of an animation or film clip, particularly, a few seconds of silence from you will be appreciated.

Most experts suggest that a slide should appear for at least one full minute. Slides with charts, tables, diagrams, or illustrations may need more explanation and should be displayed longer. This means that your 20-minute presentation should have *at most* 20 slides and perhaps significantly fewer. The rule is not absolute, and you are free to include more slides as long as you do it consciously and for a good reason.

Unless your slide contains a particularly pithy quotation, do not read from it. Find other words to amplify the idea or give additional perspective. Above all, *do not read bullet points on a slide.*

When they are done well, visual aids greatly enhance a presentation. Take the time to prepare effective visual aids.

3.3 Logistics

To avoid unpleasant surprises, practice in the actual room with the actual audio-visual equipment shortly before your presentation. Make no assumptions! It is possible that someone will change the settings of the equipment

or erase your file from the computer a few hours before the presentation. Be prepared for the network to go down just minutes before you begin.

Some students will have specialized computer demonstrations that require some work to set up. Please help us to make the transitions between speakers smooth and fast. Speak to the people who will appear before and after you and decide how you will make the shift from one person to the next. Do as much of the preliminary work as you can before the presentations begin on that day. If you have very special needs, your presentation can be scheduled first (or immediately after a break) to allow time to get ready.

Chapter 4

Prose

If you are following the project or research thesis option, this chapter is for you. It describes the paper or thesis that will be the permanent record of your work. The written work is a big part of your senior exercise. A hard copy will be archived in the Computer Science Department, and electronic copies will appear on the department web site.

If you are following the clinic option, you may skip this chapter. The written work is also a big part of your effort, but you and your teammates will write a report that follows the specifications of the Harvey Mudd College Computer Science Clinic.

4.1 Content

The purpose of the paper[1] is to explain the endeavor: its background, its goals, and its outcome. Normally, there will be an introductory chapter, a background chapter (which was drafted as the literature review in the fall semester), one or more chapters that describe the results and observations, and a conclusion. Often there will be a section on "lessons learned" or "future work." The actual organization will be determined by the nature of the work. Discuss the structure of your paper with your advisor.

Concentrate on the *results* and use them to guide the organization of the paper. Once you know where you are going, you can start at the beginning with the historical material and make your way to your conclusion. Write clearly and specifically. Know what you want to say before you begin to write. Have a concrete conclusion, and include any details you need to sup-

[1] To avoid the cumbersome phrase "project paper or research thesis," we use the word "paper" to refer generically to the written work.

port it. Present specifications, derivations, and statistical analyses whenever they are appropriate. Avoid vague generalities. Set aside time for composition. It takes work to explain a technical topic clearly and economically.

The paper should be written for other Computer Science majors: people who have a solid background in the subject but who may not know anything about the specific topic. It is especially difficult for someone who has been working on a topic for months to back off and write about the most basic and general facts. The introduction, particularly, will require patient writing and rewriting to explain the context without being tedious or long-winded.

Many students ask, "How long must a project paper be?" The answer is, "Not one word longer—or shorter—than it takes to explain the background, the project, and the results." It depends on the subject. If the project is expository or formulates concepts, the paper will be longer. If the project's goal is to create something, like a program or a robot, then the paper may be shorter. Most project papers are between 20 and 50 pages of text, not counting the front matter, appendices, and bibliography. (For comparison, this guide has 40 pages of text.) A typical research thesis is longer, but there is a wider variation among theses.

Programs or program fragments *may* appear in a paper if they are necessary for the exposition. Often, however, you can make your point just as clearly by using pseudo-code—or without reference to code at all. A full code listing is not required, even though we illustrate one in Appendix C. In some cases an advisor will ask that the code be submitted separately from the paper.

Most papers and theses will not contain a diary of daily progress. There is seldom a need for a chronography of what you did each day. The reader is interested in what you learned and why, and not in a play-by-play description of how you did it. There are a few instances, however, in which the direct experience is crucial—in projects that explore techniques of software engineering, for example—and if yours is such a project, you should describe the process.

4.2 Form

Your final document will be formatted like this guide. Most of the remainder of this chapter describes how to do it. We will use a customized style in the LaTeX document preparation system. You will have only a few decisions about the format of the paper so that you may concentrate on the prose and the content.

We expect good composition, correct spelling, and proper punctuation. Use a style manual whenever there is any question. You probably have one from your *Critical Inquiry* course in your first year, either [Hac03] or [Hac04] by Diana Hacker. *The Elements of Style* [SW99] by Strunk and White is another standard style guide. The *Chicago Manual of Style* [Chi03] is comprehensive, scholarly, and authoritative. *A Handbook for Scholars* [vL92] is a shorter work that is widely used by computer scientists, and the anthology *Mathematical Writing* [KLR96] has many good suggestions. Familiarize yourself with some of these books before you begin writing.

Find someone else—perhaps several people—to help you with proofreading throughout the writing process. Have someone read an early draft to show you where you are not clear. Later, have someone read a nearly-final draft to catch mechanical errors.

Use a spell-checker. The program `ispell` is installed on the Computer Science cluster, and it can be accessed from inside `emacs` where it will avoid many of the TEX-isms. Other systems have similar tools.

Be sure to give credit for previous work and to cite sources correctly. Most likely, you will have citations throughout your paper, and in particular, the background chapter will be dense with citations. The style of citation is described below, in Section 4.5.10.

When you are finally finished, please send copies of the completed paper to your advisor and to the instructor in charge of the senior exercise by the announced due date.

4.3 A Brief Introduction to LATEX

LATEX is a typesetting system that produces sophisticated documents. It can be very complicated, but we have tried to keep the details from becoming bothersome. It is a good idea, though, for you to become familiar with the system early in the writing process.

4.3.1 Sample Files

The directory `/common/cs/senior-exercise/latex` contains a number of files that you will find helpful.

- The file `srpaper.cls` defines the format for the senior paper; you must place a copy of it in your working directory. Should you want to investigate the inner workings of the format, however unlikely that may be, see `srpaper-doc.pdf` for the annotated source.

- For the annotated bibliography, the file `annbib-template.tex` is the place to start. It gives you the general framework for the annotated bibliography.
- For your paper or thesis, the file `srpaper-template.tex` is a place to start. It gives you the general framework for a LaTeX file.
- The source to this document is `srexercise.tex`. It provides examples of many of the constructions that you will encounter.
- The file `srpaper-sample-biblio.bib` is the bibliography file for this document. It has examples of most of the kinds of citations, including annotations, that you will want to use.

4.3.2 Resources

The standard reference for LaTeX is [Lam94], which also contains a good introduction to BibTeX. A deeper look is provided by [MGB+04]. For low-level details, see the original description [Knu84] of TeX by Knuth. Links to on-line resources and other books can be found at the following sites:

```
http://www.dci.pomona.edu/docs/tex-dci.html
http://www.latex-project.org
http://www.tug.org
```

If you are planning to write on a computer other than one in the Computer Science cluster, you may want to look at the following site which contains information about using LaTeX on Windows and Macintosh computers.

```
http://www.dci.pomona.edu/docs/tex.html
```

There are products, some of them free, that ease the burden of processing and viewing LaTeX documents. You may want to investigate them.

4.3.3 Running LaTeX

Some people execute LaTeX in a terminal window, as described below. Others prefer to use an interface program. The Macintosh computers in the Computer Science Laboratories have an easy-to-use interface program called TeXShop. If you want to install your own system, there are several different packages for each of the popular operating systems. They should all produce identical documents.

If you are new to LaTeX, start by recreating this document. On the Computer Science systems, copy these five files into the directory where you plan to keep your paper.

```
srexercise.tex
srpaper.cls
srpaper-sample-biblio.bib
pclogo-4c-noborder.pdf
```

Then simply type the command below.

```
pdflatex srexercise
```

You will create a PDF file named `srexercise.pdf`. If you are working on another system, the process may be a little different.

To begin your own annotated bibliography, start with the file `annbib-template.tex`. Copy it into your working directory, rename it, and edit it. Then use the commands `pdflatex`, `bibtex`, and `pdflatex` (twice) to create the first draft of your bibliography. See Sections 4.5.10 and 4.5.11 for more information on citations and bibliographies in LaTeX.

To begin your own literature review or paper, start with the file `srpaper-template.tex`. Copy it into your directory, rename it, and edit it. Then use the command `pdflatex` to create the first draft.

4.4 Anatomy of a LaTeX File

A LaTeX file can be created in any text editor. It is best, however, to use an editor that saves the file as plain text. We suggest that you avoid word processors because they often insert formatting information and non-standard characters.

The comment character is the percent sign `%`. A comment extends to the end of the current line.

You may want to follow along with the template or the source of this document as you read the following descriptions of the parts of a file.

4.4.1 The Class and Package Specifications

The first non-comment line of a LaTeX file is the `documentclass` specification. There are standard classes, including `article`, `letter`, `report`, and `book`. We will use a custom version called `srpaper`.

```
\documentclass[draftcopy]{srpaper}
```

The options appear between the brackets. Here, `draftcopy` is the only one. For your final copy, you *must* change `draftcopy` to `finalcopy`. That will change the pagination so that sections start on right hand pages and remove the header "Draft of ⟨date and time⟩." See Section 4.5.1 for the other options that are available.

After the class specification comes the (optional) package specifications. The packages add additional features to LaTeX and control the appearance of the output. Here are the specifications for two common packages which are used in the creation of this document.

```
\usepackage{url}
\usepackage{graphicx}
```

As you write, you may want to add other packages to the list. See Sections 4.5.2 and 4.5.4 for some commonly used packages. Do not, however, do anything to change the style or organization specified by `srpaper`.

4.4.2 The Front Matter

The `srpaper` class automatically produces a title page, abstract, and table of contents, plus optional acknowledgments, list of figures, and list of tables. The specifications for this material appears right after the class and package specifications.

The date must be of the form "Month dd, yyyy." The name(s) of your advisors must be followed with a comma and either "advisor" or "advisors," as appropriate.

Here is a minimal example of the front matter specification, with title, author, date, advisor, and abstract. When the acknowledgment is missing or empty, as it is here, the acknowledgment page will not appear.

```
\title{A Guide through the Senior Exercise}
\author{Everett L. Bull, Jr.}
\date{July 10, 2006}
\advisor{Professors Kim Bruce, Tzu-Yi Chen,
    and Sara Sood, advisors}
\abstract{A senior exercise ... }
\acknowledgment{}
```

4.4.3 The Body

Most of your paper will be ordinary text—just type it normally between the delimiters `\begin{document}` and `\end{document}`. The first line after

`\begin{document}` must be `\frontmatter` to get the title page, table of contents, and other preliminary material. A blank line divides paragraphs.

Punctuation

The most common punctuation marks—commas, colons, question marks, and the like—can be typed on the keyboard. Two matters of punctuation deserve mention, though. Quotation marks require two keystrokes each: a pair ``` `` ``` of grave accents to open the quotation and and a pair `''` of apostrophes to close it. Do not use the double quotation mark `"` from the keyboard. Some editors, including some configurations of `emacs`, will automatically translate the double quotation mark.

The keystroke `-` produces a hyphen, the punctuation mark used to break words across line boundaries and to construct compound words like "user-friendly." Two hyphens produce an en dash, used in ranges like "pages 29–47" and 2006–2007. Three hyphens produce an em dash—the punctuation mark used to set off or emphasize clauses. In math mode, a single hyphen produces a minus sign.

The keystroke `~` produces a non-breaking space. It is exactly like a normal inter-word space, except that a line-break will not occur at that point. It is normally used to connect a name and a number, as in `page~47`.

LaTeX offers a full range of punctuation marks and accents; see the LaTeX documentation for details.

Fonts

For *emphasis,* use the `\emph` command:

```
For \emph{emphasis,} use ...
```

To use italics for other purposes, the titles of books for example, use the font-changing command `\textit`:

```
Brooks wrote \textit{The Mythical Man-Month} ...
```

Table 4.1 shows the other commands for changing the shape or weight of the font.

Sectioning

Use the directives `\chapter`, `\section`, and `\subsection` to organize your paper.

italics	`\textit`
bold	`\textbf`
sans serif	`\textsf`
`mono-spaced`	`\texttt`

Table 4.1: The standard LaTeX font-changing commands.

```
\subsection{The Body}
Most of your paper ...
```

There are also subsubsections, paragraphs, and subparagraphs. To suppress the automatic numbering, put an asterisk after the directive. The heading above was created with `\subsubsection*{Sectioning}`.

4.5 Further Details about LaTeX

We do not give a comprehensive description of LaTeX. Instead, this section touches on some of the problems you are likely to encounter and gives hints at their solutions. See the references mentioned in Section 4.3.2 for more detailed information.

4.5.1 Options

The options to the `srpaper` class appear, separated by commas, between the brackets in the `\documentclass` declaration. Here are the possible options for the class.

`finalcopy` and `draftcopy` control the pagination. With the `finalcopy` option, the text is single spaced with each section starting on a right-hand page. The `draftcopy` option avoids blank pages by not skipping left-hand pages and puts a heading "Draft of ⟨date and time⟩" at the top of each page. Only one of the two options may be used. The default is `finalcopy`.

`short` is used for short papers and fragments. Use it, for example, when you are asked to submit "one chapter" of your paper. It omits most of the front matter, provides a simple title, and does not start a chapter on a new page.

`singlespace`, `onehalfspace`, and `doublespace` control the inter-line spacing. As the names indicate, `onehalfspace` and `doublespace` increase

the vertical distance between lines of text. The front matter is always single-spaced. The `onehalfspace` and `doublespace` options also make the side margins smaller—and the lines correspondingly wider. Only one of the three options may be used. The defaults are `doublespace` for `draftcopy` and `singlespace` for `finalcopy`.

`lof` and `nolof` determine the presence of a list of figures. Only one of the two options may be used, and both are ignored in the presence of `short`. The default is `lof`.

`lot` and `nolot` determine the presence of a list of tables. Only one of the two options may be used, and both are ignored in the presence of `short`. The default is `lot`.

`cm` and `mathtime` are the font options. The `cm` option gives the standard Computer Modern fonts for TEX. The `mathtime` option gives Times Roman text with matching math fonts. Special files are required for `mathtime`; see below. Only one of the two options may be used. The default is `cm`. If you choose the Times Roman fonts, you must copy all the files from the directory `/common/cs/senior-exercise/latex/belleek` into your working directory.

The documents submitted to the department must be created with the `finalcopy` option.

4.5.2 Packages

There are more packages for LATEX than you can imagine, and you will find some of them useful. You can "use" them at the top of your file, with the `\usepackage` declaration, as described in Section 4.4.1. You are free to use any package you like, as long as it does not change the overall appearance as defined in the `srpaper` class. We describe a few examples of packages here and some others in Section 4.5.4.

The `graphicx` package gives you the ability to include graphics. See Section 4.5.9. The `graphicx` package is used in this document.

The `url` package adds the facility to format url's. The `url` package is also used in this document.

The `verbatim` package adds the ability to include a file "verbatim"—in typewriter font. It is useful for code listings, as in Appendix C. One can even call in a source code file without retyping or cutting and pasting. The `alltt` package is a variant on `verbatim` that is a little more flexible. It uses

the typewriter font and respects line breaks, but it allows some of LaTeX's formatting capability. Both `verbatim` and `alltt` are automatically included by the `srpaper` class.

The `amssymb` package adds many of the less-common mathematical symbols.

4.5.3 Environments

A LaTeX environment is bracketed by the delimiters `\begin{envname}` and `\end{envname}`, where `envname` is the name of the environment. A bulleted list is created with the `itemize` environment.

```
\begin{itemize}
   \item This is the first item in the list.
   \item This is the second.
\end{itemize}
```

This code produces the list below.

- This is the first item in the list.
- This is the second.

A numbered list is created in the same way, except that the environment is named `enumerate`. The `itemize` and `enumerate` environments may be nested within one another.

Descriptions and glossaries are created with the `description` environment. The list of options in Section 4.5.1 was created with the `description` environment. The `quote` environment is handy for displaying indented blocks of text. The `center` environment does not create a list at all; it simply centers text on a line.

There are four additional environments that are provided by the `srpaper` class. The environment `indented` simply displays a block of text with the left margin shifted to the right. The environment `indented*` does the same thing and single spaces the text. If the surrounding text is already single spaced, then `indented` and `indented*` behave identically.

The environment `code` indents and shifts to the typewriter font. It is an `alltt` environment nested inside an `indented*` environment. The `vcode` environment is the same, except that the inner environment is `verbatim`. The fragment showing the `itemize` environment, above, is set using a `vcode` environment.

```
fun fib k = if k < 2
    then max(0,k)
    else fib (k-1) + (fib (k-2));
```

Figure 4.1: The Fibonacci function in ML.

4.5.4 Formatting Programs

It is difficult to display programs and code fragments in an effective and pleasing way. The most common problems in LaTeX are indentation, long lines, and reserved characters like `{`, `}`, and `\`. See the source to this document for some simple examples of short blocks of code. The construction construction that produces Figure 4.1 is particularly awkward because we want the code centered in the figure.

For short fragments of code, you can use the `alltt` or `verbatim` environments, or their indented counterparts `code` and `vcode`. For longer programs, say a full listing in an appendix, the `\verbatiminput` command from the `verbatim` package is handy. For an example, see Appendix C.

If you have formatting problems, you might look at the packages `alg`, `algorithmic`, `fancyvrb`, and `listings`. The last of these appears to parse the programming language and format it accordingly.

4.5.5 Formatting Mathematics

TeX and LaTeX were originally created for typesetting mathematics. You can include formulas like $E = mc^2$ and displayed equations like

$$\frac{\pi}{4} = \sum_{i=0}^{\infty} (-1)^i \frac{1}{2i+1}.$$

Look at the source for this document (in Appendix C) to see how these formulas were created. See [Lam94], [Knu84], and the on-line references for more about formatting mathematics.

4.5.6 Managing a Large Document

It is cumbersome to maintain a large document in a single file. You may split up your paper into several sections, each one in a different file. Use the `\input` directive to insert the contents of a file at the appropriate place.[2]

[2] We use plain TeX's `\input` instead of the more sophisticated LaTeX `\include` because the latter always starts a new page and can cause unwanted page breaks.

Frequently, there is a master file that contains only front matter and `\input` directives.

```
\input section4.tex
```

4.5.7 Labels and References

LaTeX labels provide an easy way to cross-reference material. Think of a label as an anchor. Inserting the directive `\label{name}` at some point creates a reference to the material at that point and records the page number where it appears. The reference can be an item number in an enumerated list, a section or subsection number, or the number of a figure or table. Normally the `\label` directive appears right after the declaration of the entity to which it refers. Figures and tables are a little different; the `\label` appears at the end, after the caption.

To use a label, simply type `\ref{name}`, where "name" is the name that was declared with the label. You will find it easiest, in the long run, to use descriptive names. To refer to a page, type `\pageref{name}`. Obviously, there can be only one `\label` declaration for a given name, but there can be many references to it.

Here is an example that produces the sentence "Section 4.5.7 begins on page 36."

```
\subsection{Labels and References}
\label{Subsection:LabelsAndRef}
...
Section~\ref{Subsection:LabelsAndRef} begins on
page~\pageref{Subsection:LabelsAndRef}.
```

Recall that the `~` symbol is a non-breakable space used to prevent a line break between the word "Section" and the number that follows it.

You may have to process the file twice before the labels are properly resolved. Pay attention to error messages and warnings about duplicate or non-existent labels.

4.5.8 Figures and Tables

You will almost certainly want to use some sort of figure or table in your paper. The general framework is like this:

```
\begin{figure}
      ... figure specification ...
```

```
\caption[short title, for the list of figures]{text of caption}
\label{Figure:Name}
\end{figure}
```

The order is important: first the figure, then the caption, and finally the label.

Tables are similar with the word `table` replacing `figure`. The material for a table is usually formatted with the `tabular` environment. The source for Table 4.1 is a simple example.

Positioning Figures and Tables

Figures and tables are called "floating matter" because they float among the paragraphs of the text. LaTeX's algorithm for positioning floating matter is complicated and sometimes unpredictable. Often the figure or table appears in an unwanted position.

A small change to the text—even a line or two—can drastically change the placement of figures or tables. Do not worry about the positions of figures and tables until you are ready to produce the final copy. (But do leave yourself enough time at the end to attend to the details of visual formatting!)

The first step in controlling placement is to use the placement option in the `figure` or `table` environment. The options are `t`, `b`, `h`, and `p`, standing respectively for top, bottom, here, and "on a separate page." The options appear in square brackets after the opening of the environment.

```
\begin{figure}[t]
```

Understand, however, that these options are merely suggestions; LaTeX has a number of other constraints. Sometimes it will help to add an exclamation mark, as if to say "I really mean it!"

```
\begin{figure}[t!]
```

If that does not solve the problem, there are stronger measures. Read about them in Chapter 6 of [MGB+04].

Large Figures and Tables

You may find that your figure or table will not fit within the margins, especially when using the `finalcopy` option. In those rare cases, it is permissible to have a figure or table that is wider than the text. One easy way to accomplish that is to use the `adjustwidth` environment, available with the

Figure 4.2: The Pomona College logo, as an example of included graphics.

`changepage` package. Look for the `changepage` documentation if you have a need for it.

A more drastic step is to print a wide table in landscape mode on a page by itself. The `rotating` package provides commands `\sidewaysfigure` and `\sidewaystable`.

4.5.9 Graphics

Very likely, you will want to include pictures or diagrams in your paper. You can do that with the `graphicx` package. The Pomona College logo was included in Figure 4.2 with the command

```
\includegraphics[width=0.5in]{pclogo-4c-noborder.pdf}
```

For `pdflatex`, the external graphics files must be PDF files. You will need to convert other kinds of graphics to PDF. Other variants of LaTeX allow other types of files. The book [GMR97] is a comprehensive reference for graphics in LaTeX.

4.5.10 Footnotes and Citations

Footnotes are created with `\footnote`.[3] For the senior paper we do not use footnotes or endnotes for citations. Rather, the citations are given in the text in brackets. You generate a citation with `\cite{key}`, where `key` is a name for the work cited. Here are a few examples of using `\cite`.

```
In~\cite{Cpp}, Stroustrup defined the language ...
Stroustrup~\cite{Cpp} defines the language ...
The language C++ was defined by Stroustrup~\cite{Cpp}.
```

The result is the following.

[3] Remember, though, that footnotes are distracting. Work to minimize the number of them.

In [Str97], Stroustrup defined the language ...
Stroustrup [Str97] defines the language ...
The language C++ was defined by Stroustrup [Str97].

Remember that a citation at the end of a sentence goes *before* the punctuation.

4.5.11 Bibliographies

The citation keys are associated to books and articles in a bibliography file. The file `srpaper-sample-biblio.bib` is an example of such a file. Here is a typical entry in that file.

```
@book{Cpp,
  author = "Bjarne Stroustrup",
  title = "The {C++} Programming Language",
  edition = "third",
  year = 1997,
  publisher = "Addison-Wesley",
  annotate = "The bible for a complex programming
              language. A required reference book
              for anyone doing serious C++ programming."
}
```

The entry describes the book corresponding to the key `Cpp`. Keys are not case-sensitive. The entry provides the usual bibliographic information. The sample file contains example entries for articles and other kinds of documents.

The fields in a bibliographic entry are determined by the bibliographic style you are using—a choice has been made for you by the `srpaper` class. If you add fields of your own creation, they will be silently ignored. Look at the examples, or ask your advisor, if you have an item that is not accommodated by the standard fields.

The entries in a bibliography are formatted automatically by LaTeX and BibTeX. All you have to do is provide the information in the correct form. The content of the `annotate` field is displayed when you create an annotated bibliography but not when you create an ordinary bibliography. (Technical detail: If you wish to have multi-paragraph annotations, you must use `\par` to create paragraph breaks.) See the samples files `annbib-template.tex` and `srpaper-template.tex`.

As you will see in the sample documents, the `srpaper` class provides two commands for creating bibliographies. The command `\bibliography`

creates an ordinary bibliography, like the one at the end of the document. The `annotate` field is ignored. The command `\annotatedbibliography` includes the annotations, as in the sample `annbib-template.tex`.

Only the works that you cite in the document will appear in the bibliography. Although some style guides will tell you that a bibliography must contain *only* cited works, it is sometimes desirable to include works that are not cited. The annotated bibliography is an example of such a case. To include a single work that is not cited use a command like the following somewhere in the document.

```
\nocite{MythicalManMonth}
```

To include *all* the works in a bibliography file, use the wild-card command.

```
\nocite{*}
```

The process for connecting a document to a bibliography is a little involved. First process the document with `pdflatex`. Then process the document again with `bibtex`. Then run `pdflatex` again—twice. The idea is that LaTeX gathers the keys for a citation on the first run. Then BibTeX takes the keys and formats the matching bibliography entries. LaTeX collates the entries on the second run and gets everything right on the third. If you ever add a citation or change the entry in the bibliography file, you have to repeat the cycle.

Whenever possible, refer to published journals and not to web pages. If the web page is all you have, use the format of [AKS02]. (Interestingly, that link is no longer valid. It illustrates the problems with changing web sites.) If a published paper is also available on the web, it is helpful to include the url—as was done in [AKS04].

4.5.12 Warnings and Error Messages

The messages that LaTeX displays as it runs are not always obvious, but you can usually decipher them. There is also a log file, with the extension `.log` that contains even more information.

The most common issues are overfull or underfull boxes and missing or duplicate labels. A box that is overfull by a tiny amount is perhaps not a problem, but one that runs off the page is unacceptable. All labels and citations should be correctly resolved.

When you are preparing a final copy to submit, pay attention to the warnings. Look carefully at the document and find the text that produced a warning. Make adjustments so that the paper appears exactly as you want it.

Appendix A

Sequence of Events

Precise due dates will be announced in the Senior Seminar, over e-mail, and on the departmental web site. You are responsible for knowing about—and meeting—the various deadlines. The general schedule below is provided to help you plan and balance your workload during the senior year.

There will be a few students who graduate in September or December and cannot follow the usual schedule. Those students will meet individually with the faculty and create alternate timelines.

Junior Year

Students should obtain permission for the research thesis option or the clinic option early in the second semester.

Senior Year, Fall Semester

The tasks to be completed in the fall semester are part of the Senior Seminar. The fall group progress reports will be at the scheduled time of the Senior Seminar. There will, of course, be other work for the seminar.

The due dates for the group project are not included here. There will be similar milestones that will be announced by the project's advisor.

Mid-September. Submission of a ranked list of three possible senior project advisors and corresponding project ideas.

Late September. Title, advisor, and description due. Submit the title of your project, thesis, or clinic; the name of your advisor; and a one-

or two-sentence description of the undertaking. All students will do this, even though the information will be new only for the students following the project option.

Late October. Annotated bibliography due. [Project and research thesis only.]

Mid-November Literature review and extended abstract due. [Project and research thesis only.]

Last day of classes. Revisions, if necessary, of literature review and extended abstract due. [Project and research thesis only.]

In addition, a student and advisor may agree on additional preparatory tasks to be completed during the semester.

Senior Year, Spring Semester

The schedule common to all students appears below. In addition to these activities, you and your advisor will agree on dates for activities—like partial drafts and presentation rehearsals—that are specific to your particular project.

Throughout the first half of the semester. Progress report meetings, at three or four week intervals starting in the first or second week of classes. Each student will give an informal, five-minute presentation on the project's status. Students in the group project will participate in these meetings *and* the much more frequent meetings of their project team.

Mid-April. *Complete* drafts of papers and theses due. Drafts of individual chapters will have been submitted earlier. [Project, group project, and research thesis only.]

Sometime in April, usually on a Thursday and a Friday. Presentations. The dates and times will be advertised.

One week before the end of classes. Papers and theses due. [Project, group project, and research thesis only.]

Friday after the end of classes. Senior grades due.

Sunday in mid-May. Commencement.

Appendix B

Gratuitous Advice and Crisis Management

Clearly, not every problem can be solved by reading this booklet. But answers to many common ones *can* be found here, and we have seen many problems arise when the advice and directions given here are ignored.

We expect you to read and assimilate every word of this document within twenty-four hours of receiving it. We recognize, however, that you may forget one or two details. Here are some pointers that may be helpful in a crisis. This list is expected to grow, based in large part on your experience; please make suggestions.

Getting started. If you are uncertain of which of the three options to select, read the material in Chapter 1. Take the time to make a deliberate choice from among the three options. Speak with as many people—especially faculty members and other students—as you can, and be certain that you are comfortable with your selection.

Advisor or topic. If you have selected the project option and are casting about for an advisor or topic, read Section 2.1. Speak with faculty members and other students. Think, as concretely as you can, about the nature of the project and how you will pursue it. Remember that the quality of the final product is singularly important.

Annotated bibliography. If you are faced with creating an annotated bibliography, discuss possible sources with your advisor and read Section 2.2. For help with formatting the bibliography, look at the sample files `annbib-template.tex` and `srpaper-sample-biblio.bib`, and

read about LaTeX generally in Section 4.3 and about bibliographic details in Sections 4.5.10 and 4.5.11.

Literature review. If you are about to write a literature review, speak with your advisor about the overall approach. Most likely, you have read some articles that have introductory sections reviewing previous work. These (usually) are good examples. Read Section 2.3. For the document itself, use the sample file `srpaper-template.tex`, but remember to specify the `short` option. Read Sections 4.3, 4.4, and 4.5.

Sloth. Have a specific timeline. If you are in the middle of your senior exercise and you are behind schedule, take action immediately. Reread Section 2.5. Assess the situation and make changes. Be willing to omit part of the project, if necessary. Ask you advisor to make motivational threats.

Writer's block. If you are beginning to write a paper or thesis, use a divide-and-conquer strategy. Separate the content from the form, and separate the content into manageable chapters. Discuss the organization with your advisor. Begin writing the introductory chapters early. Be sure that you have a clear vision of your conclusion. Read the first few sections of Chapter 4.

Begin writing early; there is never enough time. You should have a substantial start on your paper or thesis by the beginning of Spring Break. Writing the paper is an integral part of the project and not something to be saved until everything else is finished.

Formatting. If you are working on formatting your paper or thesis using LaTeX, review the sample file `srpaper-template.tex`. Read the later sections of Chapter 4. As problems arise, return to the relevant sections in this document, refer to the resources listed in Section 4.3.2, or ask your advisor for further help.

Pay close attention to details (*all* of them!) and produce a professional-quality document. Among other things, remember to proofread. Double check your spelling. Do not allow a line of text to run into the margin or off the edge of a page. If you do not have any tables, use the `nolot` option to suppress the list of tables after the table of contents.

Stage fright. As you plan your presentation, reread the advice in Chapter 3. Consider the visual aids. Think about the audience. Rehearse.

Then rehearse some more.

Denial. If you are following the clinic option and think that none of this applies to you, remember that you must give a presentation. See the previous suggestion.

Appendix C

Source for This Document

This appendix exists to demonstrate how to create a source listing, should you need to do it. Keep in mind, however, that the sources are usually not required. Check with your advisor.

To save paper, we have omitted the actual source listing. The following line, which would have generated the listing, was removed.

```
{\small\displayspacing\verbatiminput{srexercise.tex}}
```

The listing can be found in the directory `/common/cs/senior-exercise/latex/srexercise.tex`.

Bibliography

[AKS02] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. `http://www.cse.iitk.ac.in/users/manindra/algebra/primaliTy_original.pdf`, 2002.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in P. *Annals of Mathematics*, 160(2):781–793, 2004. Also available as `http://annals.math.princeton.edu/annals/2004/160-2/p12.xhtml`.

[Bro95] Frederick P. Brooks. *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition*. Addison-Wesley, first edition, 1995.

[Bul11] Everett L. Bull, Jr. *A Guide through the Senior Exercise*. Pomona College Computer Science Department, 2011.

[BvH82] Allan Borodin, Joachim von zur Gathen, and John E. Hopcroft. Fast parallel matrix and GCD computations. In *Proceedings of the Twenty-third Annual IEEE Symposium on Foundations of Computer Science*, pages 65–71. IEEE Computer Society, 1982.

[Chi03] *The Chicago Manual of Style*. University of Chicago Press, fifteenth edition, 2003.

[GMR97] Michel Goossens, Frank Mittelbach, and Sebastian Rahtz. *The LaTeX Graphics Companion: Illustrating Documents with TeX and Postscript*. Addison-Wesley, 1997.

[Hac03] Diana Hacker. *A Writer's Reference*. Bedford/St. Martin's, fifth edition, 2003.

[Hac04] Diana Hacker. *Rules for Writers*. Bedford/St. Martin's, fifth edition, 2004.

[Hof79] Douglas R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid*. Basic Books, 1979.

[KLR96] Donald E. Knuth, Tracy Larrabee, and Paul M. Roberts. *Mathematical Writing*. Mathematical Association of America Notes. The Mathematical Association of America, 1996.

[Knu84] Donald E. Knuth. *The TEXbook*. Addison-Wesley, 1984.

[Lam94] Leslie Lamport. *LATEX: A Document Preparation System*. Addison-Wesley, second edition, 1994.

[MGB+04] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LATEX Companion*. Addison-Wesley, second edition, 2004.

[Pom11] *The Pomona College Catalog 2011–2012*. Pomona College, 2011.

[Rab76] Michael O. Rabin. Probabilistic algorithms. In Traub [Tra76], pages 21–39.

[Str97] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley, third edition, 1997.

[SW99] William Strunk, Jr. and E. B. White. *The Elements of Style*. Longman, fourth edition, 1999.

[Tra76] Joseph F. Traub, editor. *Algorithms and Complexity: Recent Results and New Directions*. Academic Press, 1976.

[vL92] Mary-Claire van Luenen. *A Handbook for Scholars*. Oxford University Press, revised edition, 1992.

Index

The index appears here as a convenience to those using this guide. Normally, a paper or research thesis will not have an index.