CS246—A5 Project Guidelines

Your project will be graded based on correctness and completeness (60%), documentation (20%), and design (20%). Grading will proceed in two phases, outlined below:

Correctness and Completeness

The correctness and completeness component of your project will be assessed via a live demo in front of a TA. You will need to sign up for a 30-minute demo slot with a TA. Demos are typically held for 2-3 days right after the last day of classes. Sometimes, a few demo slots are also available on the last day of classes (i.e., the day the project is due). A Piazza post inviting you to select a demo slot will be made. All group members **MUST** show up, and you should plan a demo that takes the TA on a tour through all of the required features of the project. Be prepared to explain to the TA how you implemented the various parts of your project. At any time, the TA may ask you to demonstrate a particular element of your project (e.g., whether some boundary case is handled properly). If there is a component of your project that is not working, advise the TA, rather than pretend that it does.

After you have demonstrated the core requirements, you may demonstrate any additional features of the project that you may have implemented, for extra credit. The TA will assess the worth of your additional features after your demo is over.

If your project submission on Marmoset does not compile, you will not be allowed to continue with your demo. The TA will not, under any circumstance, change your code, or allow you to change your code, in order to make your submission compile. You will receive a grade of 0 on the correctness and completeness portion of the project.

Documentation and Design

The design of your project is an important part of your overall assessment. For top grades, it is not enough that your program simply work; it must demonstrate a solid object-oriented design. Thus, you will need to spend considerable time planning out your classes and their interactions, aiming to minimize coupling while maximizing cohesion. You must plan for change in your program. Imagine that the specification will change on the day before the final due date (it won't, but imagine it will). How easily can you accommodate change?

As part of your design, you should anticipate various ways your project could change. Perhaps a change in rules, or input syntax, or a whole new feature—who knows? Plan all of your project features to accommodate change, with minimal modification to your original program, and minimal recompilation. In your final design document, outline the ways in which your design accommodates new features and changes to existing features. Be specific. You may wish to implement some of these for extra credit. **This discussion is a very important component of your design grade; if you spend less than a page on it, you probably aren't saying enough.**

Your design will be assessed according to how you solved the problem. Your documentation will be graded according to how well you described your project and how well you described your de-

sign (i.e., is it clear and well-communicated?). In particular, do not expect the TA to read through your actual code. Therefore, these documents must stand alone. However, if you wish to highlight certain aspects of your design, you can indicate clearly where they can be found in your code, if the TA wants to have a look.

Extra Credit Features

All of the available projects give you the opportunity to earn up to 10% extra credit for enhancements to the core projects. Keep the following rules and guidelines in mind when planning enhancements:

- It is far more important to have your core project working well, than to have a poorly done project with extras. Remember, the enhancements are only worth 10%. Your documentation and design are also more valuable than your enhancements, so please take time to do these well.
- Your program must be able to run the core project, without enhancements, as well as with them. You are not allowed to submit two programs for grading, one with enhancements and one without. You are also not allowed to recompile your program with different compiler flags to produce versions with and without enhancements. You may select or deselect your enhancements via flag arguments on the command line. (E.g. `./project -enablebonus`, or something similar.) Even better would be if you could turn your enhancements on and off dynamically, as the program is running.
- Markers will not assign values to individual bonus features during the demo. You may show what you have done, and the TA will take note, but values will be decided once all demos are completed.
- One enhancement that is available for all three projects is offered as a challenge: complete the entire project, without leaks, and without explicitly managing your own memory. In other words, handle all memory management via STL containers such as vectors and smart pointers. If you do this, there should be no delete statements in your program at all, and very few raw pointers (the only raw pointers you would be permitted to have are those that are not meant to express ownership). Successful completion of this challenge will earn you 4 of the available 10 bonus marks. If your program leaks, these marks cannot be earned. Also note that these 4 marks are not reserved for this feature; it is theoretically possible to get all 10 bonus marks without implementing this challenge.

Due Date 1

- Submit a UML model of your system that describes how you anticipate your system to be structured. Your UML should show the classes that make up your project and the relationships between them. You only need to show public methods (i.e, you can leave out private fields and protected/private methods, unless you need to show them to illustrate a point, e.g., a design pattern). Do not show the big 5 operations, or any other constructors, accessors, or mutators. You will not be graded on the degree to which you adhere to this model, but you will be asked to account for any differences that arise between this model and your final submission. **File to Submit:** `uml.pdf`.

- Submit a brief plan of attack. Indicate what you plan to do first, what will come next, and so on. Include estimated completion dates, and which partner will be responsible for which parts of the project. You should try to stick to your plan, but you will not be graded by the degree to which you stick to it. Your initial plan should be realistic, and you will be expected to explain why you had to deviate from your plan (if you did).

  Your plan of attack must also include answers to the questions listed within the project specification itself. You should answer in terms of how you would anticipate solving these problems in your project, even though you are not strictly required to do so. If your answers turn out to be inconsistent with your final design, you will have an opportunity to submit revised answers on Due Date 2. Your plan should be no more than 5 pages long. **File to Submit:** `plan.pdf`

Due Date 2

- Submit your code. You must include a Makefile, which builds your project when you type `make`. Marmoset will attempt to build your project by saying `make`, so be sure that this does not fail. The executable should be named according to the project you choose. Depending on the projects offered this term, the names would be: `sorcery` for Sorcery, `cc3k+` for Chamber Crawler, `chess` for Chess, `watan` for Watan, `RAIInet` for RAIInet, `biquadris` for Biquadris.

- Submit a final design document. We expect that your final document will be 5–10 pages long, where each page has a word density similar to what you see in this guideline document. Organize your document into sections, using the following headings:

  - Introduction (if needed)
  - Overview (describe the overall structure of your project and an overview of aspects of your project, including how, at a high level, they were implemented.)
  - Updated UML (reflecting the actual structure of your project even if it is the same as the original UML)
  - Design (describe the specific techniques, including design patterns, you used to solve the various design challenges in the project and clearly indicate where. Also discuss how your final design differs from your design on Due Date 1)
  - Resilience to Change (describe how your design supports the possibility of various changes to the program specification. Also discuss the cohesion and coupling of your chosen program modules. Most of the design portion of your grade will be based on this summary.)
  - Answers to Questions from project (including how your answers differ from the answers you gave on Due Date 1, if they did)
  - Extra Credit Features (what you did, why they were challenging, how you solved them—if necessary)
  - Final Questions (Answers to these two questions)
    1. What lessons did this project teach you about developing software in teams? If you worked alone, what lessons did you learn about writing large programs?
    2. What would you have done differently if you had the chance to start over?
  - Conclusion (if needed)

Late Policy

We **strongly** discourage you from making last-minute changes to your code. The risk of your code not compiling, or otherwise not working, is too high. We recommend writing no new code in the last six hours before the code is due. Save those hours for preparing for your demo, and for emergency bug fixes.

Late assignments will be dealt with harshly. Assignments submitted late will be deducted 1/12 of the total mark for every hour late, rounding up. Therefore, an assignment submitted more than 11 hours late earns 0 marks. **The lateness of your project will be based on your last submission to Marmoset, whether it compiles or not.** So, before you contemplate adding a last minute feature, be sure it is worth the risk of not getting it done on time.