

CS246E—Deliverables, Project (Fall 2020)

B. Lushman

Due Date 1: Friday, November 27, 5pm
Due Date 2: Tuesday, December 15, 11:59pm

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 demo in front of a TA. You will need to sign up for a 20-minute demo slot. The demo will be done via screen sharing over MS Teams. If this is not possible for you (e.g., due to poor internet service), it is possible to have the TA mark your project without you present. In that case, you must submit a “guide” to your project, indicating the key features of your project, and any ways in which you deviated from the specification. This is to make sure the TA doesn’t miss anything as a result of you not being present. If possible, however, we recommend that you participate in your demo, as it will lead to faster marking, and likely fewer errors.

After you have demonstrated the core requirements, you may demonstrate any additional features of the project that you may have implemented, for extra credit (if your demo is not live, outline these features in your guide document). The TA will assess the worth of your additional features after your demo is over.

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. 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 documentation and design will be assessed via written documents that you will submit to Marmoset as pdfs.

Extra Credit Features

You have the opportunity to earn up to 10% extra credit for enhancements to the core project. 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 -enhanced`, 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 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 vectors and smart pointers. If you do this, there should be no delete statements in your program at all. Successful completion of this challenge will earn you 2 of the available 10 bonus marks. If your program leaks, these marks cannot be earned. Also note that these 2 marks are not reserved for this feature; it is theoretically possible to get all 10 bonus marks without implementing this challenge. (Note that this challenge is usually offered to CS 246 as well, for 4 marks. The thinking here is that you have been aware of RAII for much longer than CS 246, so this offer is more of an expectation, and not as much of a challenge.)

On Due Date 1

Submit a plan of attack. This should include a UML that describes how you anticipate your system to be structured (even if you complete the project by due date 1, the UML you submit must be one that you built prior to starting the project).

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: `uml.pdf`.

In addition, your plan of attack must include a breakdown of the project, indicating what you plan to do first, what will come next, and so on. Include estimated completion dates. 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).

If you have a partner, your document must indicate which partner will be implementing syntax highlighting, and which partner will be implementing macros. You will be required to commit to this choice.

If you do not have a partner, you will be permitted to implement both features, with the second one eligible for bonus, but the value of this feature will be somewhat diminished in comparison with a feature of similar difficulty that was not an assignment requirement for some of your classmates. Specifically, it will be valued at whatever its difficulty merits, minus one.

Finally, your initial plan of attack must include answers to all 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.

On Due Date 2

On Due Date 2, you must submit all of your code to Marmoset, together with a Makefile, such that issuing the command `make` builds your project. Your executable should be called `vm`.

In addition, you must submit a final design document. It must outline the final, actual design of your project, and how it differed from your design on Due Date 1 (if it did). You should include an updated UML, reflecting the actual structure of your project. Do this even if it is the same as the original UML.

Your document should provide an overview of all aspects of your project, including how, at a high level, they were implemented. If you made use of design patterns, clearly indicate where.

Your system should employ good object-oriented design techniques, as presented in class.

Include all answers to questions posed within the project specification, and indicate how they differ from the answers you gave on Due Date 1 (if they did).

You should not expect the TA to read through all of your code. Therefore, your design document should stand alone – the TA should not need to have your code open in order to understand your document. However, if you wish to highlight certain aspects of your design, you should indicate clearly where they can be found in your code, if the TA wants to have a look.

The most important aspect of your design document is a discussion of how your chosen design accommodates change, as described earlier in this document. You should 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.

Finally, answer the following question:

1. What would you have done differently if you had the chance to start over?

Length and Format of 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)

- Updated UML
- Design (describe the specific techniques you used to solve the various design challenges in the project)
- Resilience to Change (describe how your design supports the possibility of various changes to the program specification)
- Answers to Questions (the ones in your project specification)
- Extra Credit Features (what you did, why they were challenging, how you solved them—if necessary)
- Final Question (the last question in this document).
- Conclusion (if needed)

The document you submit will be used for both your documentation mark and your design mark. Your design will be assessed according to how you solved the problem, as described in your document, and in your Due Date 1 UML. Your documentation will be graded according to how well you described your project, how well you described your design (i.e., is it clear and well-communicated?), and your Due Date 1 document. **Together, these account for 40% of your grade, so these documents are very important.** The Due Date 2 document is worth much more than the Due Date 1 document.

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.