Battleship Peer Review

Due: at your Battleship interactive grading

We will follow up on Battleship with the peer review. You'll switch hats and review someone else's efforts for a change and provide your constructive and thoughtful feedback. The reviewed programs will be returned to their original authors. The peer review will only be graded credit/no-credit, but we do encourage you to take it seriously. If you put in quality effort, it can be an interesting and worthwhile experience— both for you and the folks you review!

What am I supposed to do?

- Read through each of the (two) peer programs you picked up (if you picked up a copy of your own, come see us to get a different one!) with a constructive and critical mindset. Don't just glance over the programs—try to understand why the code looks the way it does, and what the author was trying to do.

- Fill out the 2-page worksheet for each program. Detach the blank worksheets from the end of this handout and staple one to each of your peer programs. You will consider the program's strengths and weaknesses in areas such as decomposition, design, and readability. For each category you will answer a few questions and give a somewhat quantitative evaluation by placing a mark on the patent-pending CS106 Happy/Sad Super Slidy Fun Scale™, indicating how good you think your peer's code is in that category.

- You should feel free to give any other feedback you want to—don't feel like you have to be limited by the questions on the worksheet. Go ahead and add comments by writing directly on your peers' programs. If you like.

- Finally, you will do a similar evaluation *for your own code*, using the questions on the 1-page self-evaluation worksheet, thinking particularly about how it compares with the code you reviewed.

Bring the two completed peer worksheets, the marked up programs, and your own review sheet to your interactive grading session. The peer review feedback will be returned to the programs' authors. We encourage you to be honest, but please be constructive.

Why are we doing this exercise?

What's the point, you're asking? Well, there are several points, actually, and they're all geared towards helping you write your own code more elegantly and readably. Here are some highlights:

- It's not often that you have the opportunity to see another program that solves exactly the same problem that you yourself just did. There will certainly be areas of the program which were engineered differently—think about the tradeoffs of each approach.

- You can learn what stylistic elements contribute or detract from the readability of code. (both for your peers' code and your own code!)

- In reading someone else's code, you might find engineering and style niceties that you'd like to adopt in your own programs.

- Noticing problems in someone else's code may help you to realize similar weaknesses exist your own code.

- You'll learn what your peers (potential teammates in classes or colleagues in research and industry) think about the quality of your code. You may also receive constructive suggestions for improvement.

- It should help you to appreciate better the process that your section leader goes through when grading your assignments. (Especially of interest to those of you considering applying to be a section leader...)

Probably the most important thing for you to realize from doing this exercise is that *different people do things differently*. Different doesn't equate to better or worse—one approach is often just as valid as another. It's important to realize that the code you write in the future will almost certainly be used by other people, and those people will come with different assumptions, styles and understandings than you will. An awareness of those differences will most certainly make your code work better with other code and other people.

You are to return the two completed peer review sheets stapled to the anonymous programs, which will be passed along to the programs' authors, and your own review sheet (below). We encourage you to be honest, but please be constructive. You will not be graded on the peer review, but it will be noted if you don't do it at all.

Peer Review Worksheet

Peer's SUID:

Peer's section leader:

Reviewer's name:

In each category, mark your quantitative evaluation on the Happy/Sad Super Slidy Fun Scale™. Use the thought questions to help you brainstorm the issues and give a specific example or two to illustrate why you placed the mark where you did. Thoughtful, constructive comments appreciated!

Data Structure Design

A good data structure design is sensibly organized, complete, and efficient to access. How well does this program meet those goals? Could you easily draw a picture of the data structure? ? Would you find it straightforward to add a routine to free all the heap-allocated data? Are there modifications you might suggest to make the data structure easier to understand or modify later?

Decomposition

A well-structured program is logically decomposed into small, managable functions and has a sensible control flow. Can you easily follow the code path and understand the job of the different routines? Do you find instances of repeated code? ? If you needed to read some similar data files in another program, would you be able to easily lift some functions from this program? How difficult would it be to add new functionality to the game, say, for instance, allowing the user to place the ships?

Maintainability

Code is often inherited by someone else for bug-fixing and adding new features, so it's important the original author take time for good design, clean coding, and insightful comments. If there were a bug in this program, how hard would it be for you to find and fix? ? Would you feel at ease changing the program to allow ships to be placed diagonally? Do you find the comments too sparse, too verbose, too fluffy, just right? What would you change to make this code more maintainable?

Readability & Style

Well-chosen variable and function names, symbolic constants, consistent indentation, and appropriate use of white space, among other things, can greatly enhance the readability of a program. Does the program have a consistent style? Are the identifier names informative and accurate? Do you find the program aesthetically pleasing to read? If you were to inherit this code, would you want to make a lot of stylistic changes?

In Conclusion

What are the two best things you noticed about your peer's code?

What are the two things that bothered you the most about your peer’s code?

Peer Review Worksheet

Peer's SUID:

Peer's section leader:

Reviewer's name:

In each category, mark your quantitative evaluation on the Happy/Sad Super Slidy Fun Scale™. Use the thought questions to help you brainstorm the issues and give a specific example or two to illustrate why you placed the mark where you did. Thoughtful, constructive comments appreciated!

Data Structure Design

A good data structure design is sensibly organized, complete, and efficient to access. How well does this program meet those goals? Could you easily draw a picture of the data structure? ? Would you find it straightforward to add a routine to free all the heap-allocated data? Are there modifications you might suggest to make the data structure easier to understand or modify later?

Decomposition

A well-structured program is logically decomposed into small, managable functions and has a sensible control flow. Can you easily follow the code path and understand the job of the different routines? Do you find instances of repeated code? ? If you needed to read some similar data files in another program, would you be able to easily lift some functions from this program? How difficult would it be to add new functionality to the game, say, for instance, allowing the user to place the ships?

Maintainability

Code is often inherited by someone else for bug-fixing and adding new features, so it's important the original author take time for good design, clean coding, and insightful comments. If there were a bug in this program, how hard would it be for you to find and fix? ? Would you feel at ease changing the program to allow ships to be placed diagonally? Do you find the comments too sparse, too verbose, too fluffy, just right? What would you change to make this code more maintainable?

Readability & Style

Well-chosen variable and function names, symbolic constants, consistent indentation, and appropriate use of white space, among other things, can greatly enhance the readability of a program. Does the program have a consistent style? Are the identifier names informative and accurate? Do you find the program aesthetically pleasing to read? If you were to inherit this code, would you want to make a lot of stylistic changes?

In Conclusion

What are the two best things you noticed about your peer's code?

What are the two things that bothered you the most about your peer’s code?

Self Evaluation Worksheet

After you've had the chance to take a look at two other people's code, we'd like you to reflect a little bit on your own code. We'll start you off with the same semi-quantitative evaluations you did for the peer code:

Data Structure Design 

Decomposition 

Maintainability 

Readability & Style 

What do you feel are the two best things about your code?

What two things about your code would you change now that you've reviewed someone else's?