Program Critique

Before doing the write up for the code review, attempt to make the following change top the project. **Spend no more than 90 minutes attempting the change.** If you can't get all of the changes done in 90 minutes, stop. Doing this will give you an insight into the program and how it was written. This will allow you to make better comments when you have to write up your review.

> At some time in the past, a "standard deck of cards" had 5 suits instead of the 4 suits we see today. This fifth suit sometimes used a circle as the suit design was called "Circles" or "Balls". Another common design for the fifth suit was a star using the name of "Stars". Since the letters S and C and already in use for the suits of Spades and Clubs, the following discussion will use the name of "Balls".
>
> For the code you are to review, increase the deck of cards to contain 5 suits: Balls, Clubs, Diamonds, Hearts and Spades. Each suit has the same 13 ranks as originally defined. This will give the deck a total of 67 cards. This will cause only one change in the game play, hands can now be evaluated to have "Five of a Kind" which is the absolute best hand a player can have. Five of a Kind is very similar to Four of a Kind, except all 5 cards must have the same rank.
>
> So the changes you are to attempt is to:
>
> 1. add the fifth suit, the suit of Balls, to give the deck 67 different cards and
> 2. add the evaluation of "Five of a Kind"

Part 1: Who is Critiquing Whose Code?

Your write up must first have the following information clearly expressed at the top of your critique. This information is needed to help get the proper information to the correct student.

1. The number of the project you are critiquing.
2. The name of the student whose code you are critiquing.
3. Your name.

An example of this information is as follows:

```
Critique of Programming Project 1
Code written by for Hong Cao, hcao14@uic.edu
Critique written by Amit Bhadoria, abhado8@uic.edu
```

In the above information, Amit Bhadoria wrote the critique and Hong Cao wrote the program. Your name and user ID is to replace those of Amit Bhadoria.

Part 2: How Successful was the Modification

The second part of your critique is to describe how far along you got with the above changes in 90 minutes. Where you able to complete all of the changes? Where you only able to add the 5th

suit but not evaluate Five-of-a-Kind? etc. Was there something already existing in the code that either aided or hindered your ability to make changes?

Describe how easy or difficult is was to understand and make the needed changes.

Part 3: The Actual Critique

For each section of the critique, you are to write comments about your opinion of the code you are to critique. You are to give supporting comments of your opinion with specific examples from the code you are critiquing. Simply giving yes/no answers will not be accepted as correctly doing the critique. Also you are to rate each area with a score from 1 to 5. Where 1 means the area was poorly addressed in the program and 5 means the area was addressed very well in the program. Thus, each section of the critique is to contain:

- a score from 1 to 5
- an explanation for the given score
- specific examples from the original code or design document that highlight the points made in the explanation. These example could come from your experience in making the above change or the example could come from other things you noticed in the program.

Use the information from the Generic Checklist for Code Reviews for additional ideas to look for when answering the following questions. Note that Generic Checklist does not organize its points in the same way as presented below.

The 5 Critique Sections

Your critique is to address the following 5 areas. Each area must have its own section in your write-up. The questions listed are to give you an idea of what types of things you are to look for (and write about) in your critique.

3.1 - Supporting Documents (Review of the none code parts of the project)

How well do the supporting documents introduce you to what is going on in the program? Are they understandable (use well written sentences and proper grammatical form)? Does it have a file header comment at the beginning of every file? Note that this could include readMe files and program block comments that describe the overall program.

3.2 - Program Style (Review of non-programming features of the code)

How well is the program commented? Does every function have a function header? Does it make good use of in-line comments? Are identifiers given meaning names? Does the program use indentation and blank lines in a uniform manner to help make the program easier to read?

Use the Google Code Style Guide as a base for good programming style:

- https://google.github.io/styleguide/javaguide.html

Also the Generic Checklist for Code Reviews gives idea to check for with respect to Programming Style.

3. 3 - Object Decomposition (High level object review)

How well are the various Objects in the project developed? Do the objects properly break down the application domain? Do they seem to follow "good Object Oriented practices"? Does the code use inheritance when appropriate? Do the data members of the objects adequately represent the information needed by the object? Do the objects/data structures properly model their "real world" entity?

Does the code make proper use of **cohesion** and **coupling**?

- The best use of **cohesion** is when each function performs one well defined operation. Are the functions of small enough size to be easily understood (or are there a few huge functions that do everything)?
- The best use of **coupling** is when each function does not depend directly on the operations performed by other function. This is reached by maximizing the use of parameters and minimizing the use of global variables. If changes were made in one function/method, would other parts of the program be affected?

3.4 - Use of Data Structures and Algorithms (Internal Review of Objects)

Do the objects contain data structures that use memory efficiently? Do the algorithms use machine cycles (time) efficiently? How do the data structures and algorithms used compare to the data structures and algorithms used by your program? Are decision statements (if, while, etc.) easy to understand? Was the code written to help minimize errors?

3.5 – Program Structure (General Coding Techniques)

Was the structure of the code written in a way that made the code easy to read and understand? Did the program make use of multiple source code files? Did each source code file contain the "proper" information? Does the program make use of modern programming practices? (I.E. exception handling) Look for items in the Generic Checklist for Code Reviews for program structure ideas.

Part 4 – Summary/Overall View of the Project

Summarize your comments about the code. Give an overall rating from 1 to 5 on the code. If you were to make additional changes to this code, do you think you could do so in a "timely manner"? Why or why not?