### A10b

When you submit your code file for A10a, you will be asked to do two code reviews of other submissions. We discussed code reviews in Lecture 17, and did an in-person code review in lab that week. Here you will provide two written code reviews. Code reviews are related to refactoring in that the reviewer can make suggestions about how to make the code clearer.

If you do not have two other submissions to review let me know. At a certain point I need to prompt the system to assign the last few reviews.

The review cannot be some generic statement like "well done!", or "good variable names!". You should carefully read the code, investigate anything you do not completely understand, then make specific and actionable suggestions.

Here is what you should do for the review:

Look closely at the code for the first few functions (the binary search is not likely to be very different for different people, and the testing, while important, is a different topic).

* 1. Look at the variable names used in each function. Do they describe the kind of values they hold? Did you have to look at the code to see how it was used to understand the name? If not, what would be a better name, and why?
  2. Look at the formatting of the code. Are there unnecessary things like extra ()? Is there consistent spacing, like a space around an operator or after a comma? Do variables match Python style of lowercase words with a \_ between words? Make suggestions as needed. Reference specific lines of code, not just "add spaces where needed".
  3. Look at the use of Python. Are there things that seem unneeded, like two variables each storing the same data? Are there if statements checking things that do not need to be checked? Are things repeated that do not need repeating? Do loops use the most straightforward form (index for location, value for just values). Is append or + used appropriately? Describe where you see an issue and how it can be simplified or improved.
  4. Look back at your own assignment. If something is done in a different way than you did, think about which way you think is better and why. Write your thoughts.

For each review, make a numbered list 1-4, and write a few sentences for each number matching the points above. Make sure they discuss specific things in the code being reviewed, and make suggestions for improving those things. Then, add a few sentences with more of a summary of the overall code.

If you cannot find something to comment on for one of the topics, you can explain why you felt it was so well done. I will be looking to see if I agree there is nothing to be improved.

The review should be written in the Canvas text boxes provided for peer review. They expand as needed.

Take care to write complete English sentences with spelling and punctuation. Peer reviews that are mean, snarky, demeaning, etc. will receive little credit. You should attempt to create understanding.

Your grade for this portion will be assigned in A10b, but there is no submission for A10b except to do the peer reviews here in A10a.

### Example Review of a Pretend Assignment

1. In the shuffle\_cards function, there is a variable cds that stores a dictionary of playing cards. The name deck, or unshuffled\_cards would explain its purpose more clearly. Then, numc is used to count the number of cards. A variable name saying that purpose, like number\_of\_cards, lets the reader understand the expected value stored in that variable.

2. Several variables are spelled in all caps, like CHANGINGVARIABLENAMETOAVOIDPLAGIARISMDETECTOR. Python style is to use lowercase words separated by an underscore. One variable name is misspelled - you can refactor numbre to number using refactor->rename in Pycharm. When you call the range function, sometimes you have a space in the parentheses and sometimes you do not, like range( 10) or range(10). Remove the extra spaces.

3. In the loop to count the score, you check each time if the total score is 0. That can be done once before the loop starts.

4. We used different approaches to find the winner. I wrote it with two if statements, and you used an if-else form. Your else avoids writing the condition again, which is nice.

Overall, you mostly used clear variable names, and your code solves problems without extra unneeded steps.

**A10b Rubric**

| **Criteria** | **Ratings** | **Pts** |  |
| --- | --- | --- | --- |
| This criterion is linked to a Learning Outcome  Do the reviews make specific comments on variable name choices? |  | 20 pts |  |
| This criterion is linked to a Learning Outcome  Is the code format, spacing, and naming style specifically addressed with specific examples? |  | 20 pts |  |
| This criterion is linked to a Learning Outcome  Is code logic and elegance discussed in specific ways? |  | 15 pts |  |
| This criterion is linked to a Learning Outcome  Does the reviewer discuss differences in approach between their own solution and the solution being reviewed? |  | 15 pts |  |
| This criterion is linked to a Learning Outcome  Are the reviews written with reasonable effort for clear written communication and in a tone appropriate for constructive criticism? |  | 30 pts |  |
| Total Points: 100 | | | |