SP20 Comments on Prelim 1, statistics, and regrade instruction

Please **read the comments below on each question** of the prelim to learn what are some of the common errors (to avoid in the future) and what are the concepts that you may need to review. If you have not done well, be sure to catch up and then you can show us at the next exams what you have learned! *Remember, Prelim 1 is worth only 20% so you have plenty of opportunities to improve.*

Please do NOT just read the solutions! Re-do the problems without reading the solution in order to LEARN.

Please talk to Dr. Fan and Dr. Muhlberger if you have questions about catching up or about your status in the course.

Regrade Requests

If you see a *grading error*--the grader mistook your correct code to be incorrect and deducted points--then please submit a regrade request on Gradescope and specify exactly what and where the grading error is. On the other hand, do not ask for a regrade if you dislike the number of points deducted--the same grading rubric was applied for all students and will not be changed.

You can submit a Prelim 1 regrade request on Gradescope *until 11 AM (EDT) Monday, April 13*.

Statistics

Max:	99
Median:	81.0
Mean:	77.4
S.Dev:	16.1

```
% Action to take after prelim
if  yourScore > 86

        toDo= celebrate + later look at the solutions

elseif  yourScore > 60

        toDo= redo problems + later look at the solutions

else
        toDo= meet with course staff to go over exam …
                + re-do the problems--do NOT just read the solutions ...
                + stay caught up from now on
end
```

Question 1: Traces (a) for-loop and accumulation pattern, conditional, remainder, print format; (b) function scope

- (a) Good job tracing through the calculations - most students correctly figured out which numbers would be displayed
- (a) But be careful with output formatting - the most common error was not right-aligning the output. Other formatting errors were common as well; details do matter in engineering.
- (b) Function scope: mostly well done! If you got this part wrong, be sure to discuss it with a member of the course staff. **You must understand scope perfectly** in order to deal with a later topic in the course (recursion).
- (b) For the third line printed, many students said "a is 100"; perhaps you were confused by the line a = b^2 in the function and accidentally set a=a^2(=10^2).

Question 2: (a) loop guard and checking input; (b) random integer generation; (c) linspace and range expressions (continuous-->discrete)

2a:

- Most students correctly identified the comparisons that would indicate that a point was outside of the shaded region, and many even handled the boundary case correctly
- However, even with correct clauses, many students combined them with the wrong logical operator, effectively saying "while both coordinates are outside" instead of "while either coordinate is outside". Always double-check your AND vs. OR, especially when inverting an "until" into a "while" condition.

2b:

- Most students got this correct despite the tricky 2D context - good job!
- Among incorrect answers, many students missed the scaling when generating random integers. Might be worth going through each step of the generation process: rand() generates a random number within (0,1), a*rand() gives a random number within (0,a), and thus floor(a*rand()) gives a random integer within [0, a-1], not [0, a].
- Unfortunately, if you get the scaling wrong, there's a good chance you'll also pick the wrong shift (since one of your boundary cases will be off)
- Some students didn't use the ceil() or floor() function to round the random number to an integer. The circles in the plot were specified to be at integer positions, and rand() doesn't automatically generate an integer.

2c:

- Many students used n as the step size, but for ranges, the middle argument is a step size, not the number of elements

- A very common mistake was to use n as the denominator of the step size, rather than (n-1). Recall that when counting tick marks (for example), there's always one more tick than there are spaces between them, and to get the step size, you need to divide the width of the interval by the number of spaces (steps).

Question 3: (a) function with multiple outputs (b) calling a function with multiple returned values, algorithm for finding the best in a set with constraints

3a:

- Good job on the calculations! Only a handful of mistakes there
- Remember the difference between a script and a function - a function *returns* its output (no print statements), and its inputs are provided through *parameters*, not input() statements

3b:

- Many students used a while-loop instead of nested for-loops. In this problem, although there are conditionals to be checked, you still have a fixed number of iterations to go through, so using for-loops will be the most straightforward way of solving this.
- Many used < instead of <= when checking the conditions. Read the English requirements carefully; e.g. "cannot exceed/cannot differ by more than," and double-check the equality case if you need to invert the logic.
- Cost needed to be constrained in both directions (not too expensive and not too cheap) - again, read the specifications carefully.
- When calling a function, be sure to assign its outputs to variables of your own. When in doubt, you can always check back at the function definition for the required inputs and outputs.
- Some students used a single while-loop approach where the two parameters are incremented simultaneously. Not all combinations are checked.

Question 4: function header, indefinite iteration, conditional statements, using probability in simulation, building a vector with indefinite iteration, calling a function

4a:

- Most students applied the correct accumulation formulas to update the hero's and monster's hitpoints depending on the action taken - good job!
- Additionally, most students got the condition for the while loop correct to check if the monster and hero were still alive
- And most people implemented the array index correctly (both the initialization and the accumulation)

- The input parameter to the function should be used as a threshold for whether the hero attacks or heals. Many students either 1) compared this threshold to 0.5, rather than a random number, meaning that only one branch of the conditional was ever reached, or 2) did not use the input parameter and instead compared the random number to 0.5
- Many students had the monster attack regardless of whether it was alive or not! Since the hero always has a turn before the monster, it is possible that the monster dies before it gets a chance to attack back. This should be checked in a separate if-statement.
- Many students had the monster attack before the hero, rather than the other way around as indicated in the problem statement.
- Some students did not append to the hps array with every round, only after certain actions. For example, some students didn't append to the array when the hero's action was to heal. The length of the returned array needs to match the number of rounds, regardless of which "if" branches are executed inside the loop body.

4b:

- Most students knew to check if the hitpoints were greater than zero to determine if the hero was still alive
- Many students tried to access variables that are local to the simCombat() function and are not available in the calling script, especially the monster's hitpoints
- Students often checked if hps > 0 (the entire array is greater than zero) rather than hps(length(hps))>0 (the final hero hitpoint value is greater than zero)

Question 5: problem decomposition, nested loops, linear interpolation (and perhaps accumulation pattern)

5.

- Students did a good job of looping through the correct number of rows (either with a for- or while-loop), and got the number of bricks in each row correct as well. Almost everyone set up nested for-loops or a for-loop within a while-loop.
- Most students called DrawRect() with the 5 input parameters in the correct order, (though occasionally the values of the arguments were not correct)
- Most students at least increased the brightness of the bricks going from left to right, even if the exact color value was not correct.
- Some students did not shift the x-coordinate of the bricks properly within a row. Each brick should be moved right of the previous brick by a distance of w (width). Instead, we saw students shifting bricks by w*row_number
- A lot of students had uninitialized variables, such as not setting x0 and y0.
- The color of a brick depends only on the x coordinate of the midpoint, normalized by the total width n*w. But many students instead tried to use the relative position of a brick in a row (something like c/(n-r+1)). Don't try to get too tricky with the math - if the problem mentions a quantity like "midpoint," try calculating and using that quantity.

- The problem specified the coordinates of the top-left corner of a rectangle, but DrawRect() takes as arguments the coordinates of the bottom-left corner. Many students failed to adjust for this difference.
- Some students called DrawRect() in the outer for-loop, which would only draw one brick per row (or one extra brick if it's also called in the inner loop)
- Some students did not reset x after plotting an entire row of bricks, so the next row starts from the right of where the previous row ended.