CS100M Spring 2007: Project 2 Grading Guide

The coded items below (e.g., c1e, s2a) indicate what a student's solution should accomplish. Codes that begin with the letter 'c' deal with correctness; codes that begin with 's' deal with style.

Grader: If a student's solution does not accomplish task c1a, for example, then write the task code 'c1a' along with any diagnostic remarks you can give. Count the number of correctness and style errors separately.

Items marked with ** count as two errors. In the table below, the top row lists the possible scores (1 to 5). The next row lists the number of correctness errors corresponding to every score category. The style score is determined similarly. Enter the total score (maximum of 10) in CMS as the project score. If there are bonus questions, enter any bonus points separately in the "Bonus Bucket," separate from the project score.

Student: Read the grading guide for every project, even if you get a perfect score! Notice from the table below that we often give one or two "freebies," i.e., mistakes that don't cost you any points. Learn from working on the project, and learn from any mistakes.

Scores

- c and s stand for correctness and style; see table below.
- parts with ** next to them means that they are double the value, *** for triple, etc.
- Apply bonus for exemplary work or doing additional tasks.

Score	0	1	2	3	4	5
#correctness errors	>9	7-9	5,6	4	2,3	0,1
# style errors	>10	8-10	5-7	3,4	2	0,1

General

(s0a) Use meaningful variable names
(s0b) Appropriate indentation
(s0c) Appropriate comment header in each script/function file
(s0d) Appropriate and concise comments throughout
(s0e) Reasonable line lengths; no horizontal scrolling
(s0f) [up to **] No superfluous code
(s0h) No debugging output.
(c0a) [2* max] Program compiles without error. (1 * for each compiler error message up to 2)
(c0b) [2* max] Program successfully executes without crashing. (* for occasional, ** for persistent)
(c0c) [2* max] Retrieve information from the user correctly

(s0g) Don not use arrays in this project, as described in the project specification.

1. Rolling Dice

(c1a) correctly simulate two dice with random numbers:

ceil(6*rand(1)) + ceil(6*rand(1))

(c1b) correctly stop the program if the first roll is 2, 3, 7, 11, or 12
(c1c) correctly use a loop to control the rolls in oneTurn.m
(c1d) correctly call oneTurn.m inside a loop
(c1e) get the value correctly after calling oneTurn
(c1g) do the sum and averaging correctly

(s1a) format the output in each turn reasonably
(s1b) format the output of the average reasonably

2. Approximation via Taylor Series

(c2a) correctly control and stop the summation of the terms with a while loop
(c2b) calculate the terms correctly
(c2c) count the number of terms correctly
(c2d) correctly use the exp function to get the exact number

(s2a) should use one while loop instead of nested loops
(s2b) displaying the output reasonably

3. Twinkle, Twinkle Little Star

(c3a) draw all the points and lines
(c3b) correctly calculate the x and y coordinates in function angle2xy
(c3c) correctly pass return values from function angle2xy
(c3d) correctly draw the star shape according to different increment
(c3e) implement star as a function and don't ask users for input

(s3a) the resulting picture is clear