CS100M Spring 2007: Project 3 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**
Number of correctness errors	> 11	8 – 10	6 – 7	4 -5	2 – 3	0 -1
Number of style errors	> 9	7 - 8	5 - 6	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

Part 1: integral.m

(c1a) correctly calculates h
(c1b) correctly calculates value
(c1c) correctly uses cumsum and vector operations in integralappr1
(c1d) correctly uses cumsum and vector operations in integralappr2
(c1e) correctly uses cumsum and vector operations in integralappr3
(c1f) correctly uses loop-bounds in integralappr1
(c1g) correctly uses loop-bounds in integralappr2
(c1h) correctly uses loop-bounds in integralappr3
(c1i) correctly calculates the value integralappr3

Part 2: checkvictory.m

(c2a) checks for vertical arrangement
(c2b) checks for horizontal arrangement
(c2c) checks for diagonal down right arrangement
(c2d) checks for diagonal down left arrangement
(c2e) function does not try to access elements outside the array
(c2f) function correctly returns the winner
(s2a) function should test for victory using <condition> && <condition> && ...

Part 3: findbottom.m

(c3a) correctly returns the lowest possible empty space
(c3b) code tried to excess elements outside of the grid
(s3a) only one while loop necessary

Part 4: connect4.m

(c4a) program correctly detects which column is clicked even when click is outside grid
(c4b) correctly detect when there is no more space in the column
(c4c) correctly updates "grid" matrix
(c4d) correctly calls drawx() to draw the appropriate color crosses
(c4e) correctly switches between the two players
(c4f) correctly updates title message
(c4g) program ends when there is a winner or grid is full
(s4a) student does not edit code outside of allocated portions **except for grid(rows,cols)**