CS100M Spring 2008

Grading Guide: Project 6

May 6, 2008

The coded items below (e.g., c1e, s2a) indicate what a student's solution should accomplish. Codes that begin with the letter "c" deals with correctness; codes that begin with "s" deals 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 ands stand for correctness and style; see table below.
- parts with ** next to them means that they are double the value, *** for triple, etc.

Score for part 1		2	1	0
#correctness errors		0-1	2-3	4+
#style errors		0	1-2	3+

General

- (s0a) Use meaningful variable names
- (s0b) Appropriate indentation
- (s0c) Appropriate and concise comments throughout.
- (s0d) [up to 1* per m-file] Comment header is *required*.
- (s0e) Reasonable line lengths; no horizontal scrolling
- (s0f) [up to **] No superfluous code
- (s0g) [up to **] Reasonably efficient and concise code; a little inefficient is OK

- (s0h) No debugging output
- (c0a) [2* max]Program successfully executes without crashing.(*for occasional,**for persistent)
- (c0b) [up to 1* per m-file] Functions are defined as specified by the project (parameters are of correct type in correct order)

1 H-Tree

- c1a : Student attempts to use recursion to solve the problem.
- c1b : Each individual call of function drawHTree plots an H correctly.
- c1c (up to 3*) : Recursive call is correct. (one point for correct position (x,y) of new H, one point for correct update of length/width, one point for correct update of level)
- c1d : Correct termination of recursion (4 or 5 levels of drawing accepted, but no less and no more).
- c1e (up to 2*): Correct implementation of color scheme: (*)90% of H's are blue, 10% not blue, (*) of the remaining 10%, at least four colors are randomly chosen from.
- s1a : Recursion is used in an appropriate manner.
- s1b : Efficiency: no unnecessary (repeated) recursions.

2 Decode a phone number

Because this project is so open ended, the grading guide is a little different. Grades are divided into three categories: technical requirements, programming style, and report. A student may earn 0,1, or 2 points in each category for a max score of 6 on this section.

Technical Requirements:
0 – Program fails to accomplish the task (fails on all test cases) OR noise level handled is <20%.
1 – Program passes all test cases up through 20% noise level OR program takes longer than 5 minutes to run a single test case. (see note)
2 – Program passes all test cases (handles 50% noise level).

Programming Style:
0 – Any of the following:
– Commenting is not attempted.
– Code is designed very haphazardly (no attempt to divide functionality into sections or subfunctions).
– Code is incomplete
1 – Any of the following:
– Commenting has been attempted, but is sparse or incomplete (missing function headers for example).
– Code design may be messy or show evidence of a lot of last-minute hacking.
– Any other style error not mentioned thus far (see "General" - ex: debugging output, no indentation, long lines, etc)
2 – User can glance through the code and get a sense of what is going on with little effort. Code is well layed out and evidence of planned design is seen.

Report:
0 – Any of the following:
– Report has not been attempted.
– Report does not address the issues specified in the project (% noise and number of digits handled) .
– Report vastly overestimates performance of student's code.

– Report demonstrates student did not test their code (very large disagreement between grader's testing and report).

1 – Any of the following:

– Report addresses the issues specified in the project, but may be unclear or poorly written.

– Report shows lack of thorough testing (some disagreement between grader's testing and report).

– Program takes longer than 5 minutes to run a single test case and no mention is made in report.

2 – Report is clearly written, addresses the issues mentioned, and thorough testing is shown (report does not disagree with grader's testing within a small margin of error (a few %)).

NOTE: Any student who lost points due to inefficient code (took longer than five minutes to run a test case) may make up lost points by emailing TA Alex at ajd45@cornell.edu to receive a set of test cases. The student should run their code on these test cases and write a short report which includes the following: results on the test cases (returned phone numbers - list form is fine, or any errors encountered), a short write-up explaining why their code took so long to run and a suggestion for how to fix it (one paragraph should do it). The student should submit this report as part of a regrade request at any TA's office hours.