CS100M Fall 2007

Grading Guide: Project 3

March 7, 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	5	4	3	2	1	0
#correctness errors	0 – 1	2 – 4	5 – 8	9 – 13	14 – 19	20+
#style errors	0 – 2	3 – 6	7 – 10	11 – 15	16 – 20	21+

General

- (s0a) Use meaningful variable names
- (s0b) Appropriate indentation
- (s0c), (s0d) [up to 6*, one for each m-file] Appropriate and concise comments throughout. 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)

1 Chaos Game

1.1 chaos.m

1. (c1a) Correctly uses the vertex values (i.e. $[0\,1\,.5], [0\,0\,\frac{\sqrt{3}}{2}]$) in chaos.m
2. (c1b) Randomly generates selects one of the three vertexes
3. (c1c) Correctly generates the middle point from current location to vertex

1.2 chaosGame.m

1. (c1d) Correctly allocate space for holding data set
   Note: OK to start from fixed location since question was worded so....
2. (c1e) Correctly iterate over n points
3. (c1f) Set axis equal off
4. (c1g) Should work for n = 6500, appropriate figure should appear
5. (s1a) Use appropriate colors so that the graphics are clearly visible
6. Note: (s0g) If the plot occured *inside* the loop, that is inefficient

2 Testing the Collatz Conjecture

2.1 Collatz.m

1. (c2a) Correctly uses the while (no break, no for loop) loop to check for the termination condition ($num \neq 1$)
2. (c2b) [2*max, 1* for each mistake found] Correctly uses the if statement to check for the even and the odd cases, and uses the right equation for each case.
3. (c2c) Correctly forms and *returns* the output vector.

2.2 tryCollatz.m

1. (c2d) [2*max, 1* for each mistake found] Correctly uses a loop to take repeated inputs from the user and uses it to call collatz. Uses the right termination condition to stop the intake of inputs
2. (c2e) Correctly uses the length function (or the appropriate size function) to print the length of the vector
3. (c2f) Correctly prints the sequence (printing has to occur here)
4. (s2a) While taking user input, tells the user that a '0' input is used to quit. Displays the output neatly.

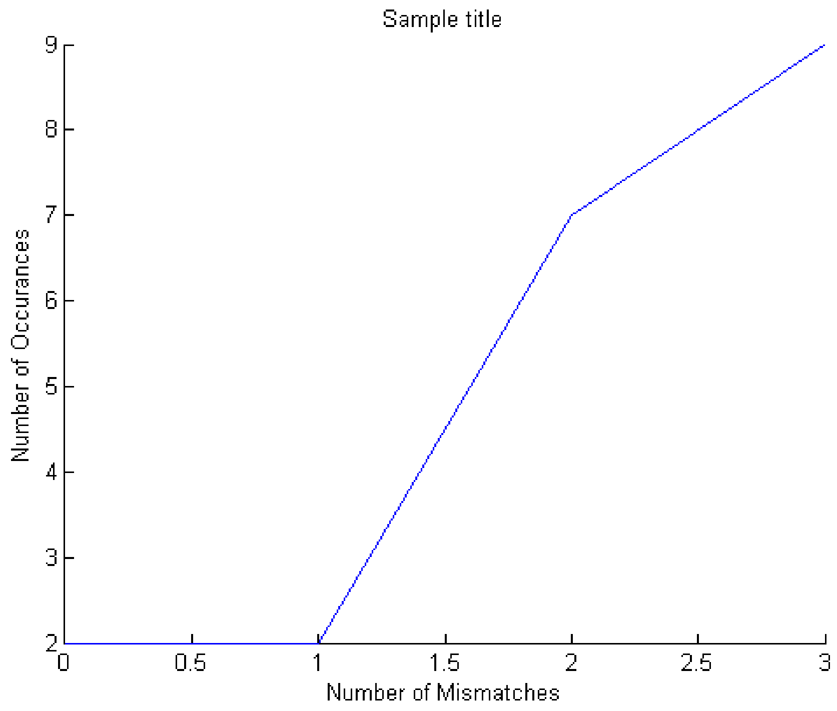
3 Searching for Similar Substrings

3.1 partialMatches.m

1. (c3a) Loop: repeatedly aligns the query with a substring of the sequence
2. (c3b) Correctly makes a comparison between the two (sub)strings– can be done as an inner loop or use vectorized code.
3. (c3c) [2*max, 1* for each mistake found] Accounts for mismatches
4. (s3a) Function *returns* a vector.

3.2 plotMatchData.m

1. (c3d) Makes an x-y plot or histogram or bar graph (plot needs to have lines, not just a dot for each value)
2. (c3e) Values for x and y are correct
3. (c3f) y value is assigned correctly with a function call to partialMatches
4. (s3b) axis are labeled is required (title is optional)
5. (s3c) plotMatchData('mississippi', 'iss'); produces the following;


Sample title
Number of Occurances
Number of Mismatches