CS100M Spring 2006: Project 5 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.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.

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.

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

General

(s0a) Use meaningful variable names
(s0b) Appropriate indentation
(s0c) Appropriate comment header in each function file
(s0d) Appropriate and concise comments throughout
(s0e) Reasonable line lengths; no horizontal scrolling
(s0f) No superfluous code
(s0h) No debugging output.
(s0i**) Submitting class files rather than source files.

(c0a) [2* max] Program compiles without error. (Grader: please try to to fix obvious cases.)
(c0b) [2* max] Program successfully executes without crashing. (* for occasional, ** for persistent)
(c0c) [up to 3*] Preserves given code.

api.txt

Subtract 3 correctness points if not submitted at all.

Complex.java

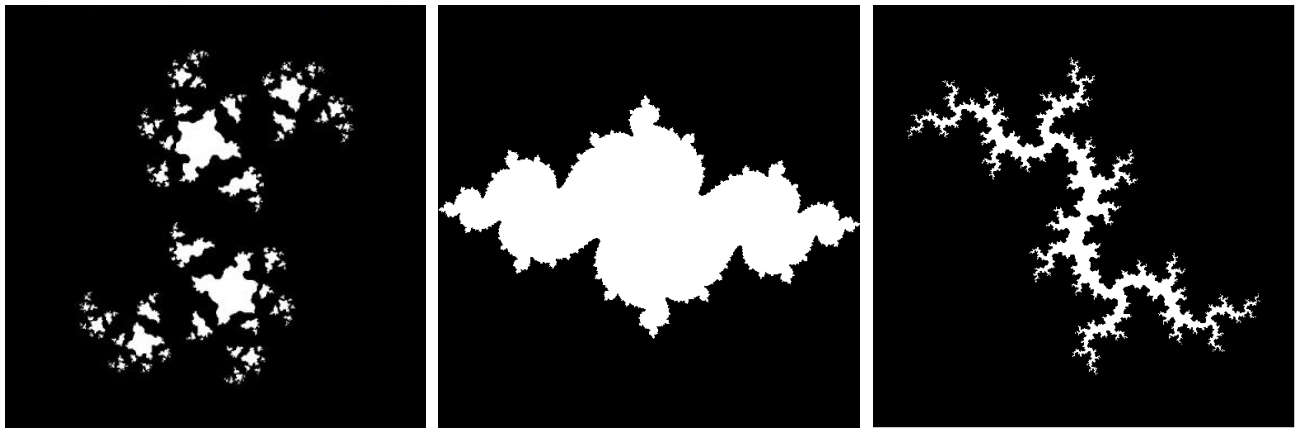
(c1a) First constructor is correct.
(c1b) Second constructor is correct.
(c1c) Subtraction is correct.
(c1d) Multiplication is correct.
(c1e) Division is correct. (Division by zero does not have to be checked.)
(c1f) Magnitude is correct.
(c1g) Angle is correct.

(c1h) toString() is correct. (Spaces optional.)

Fractal.java

(c2a) [up to 2*] Correctly implemented looping and prompt.
(c2b) [up to 2*] Correctly implemented isInsideFractal.

These are the fractals for c = 0.4 – 0.4i, c = -0.75 + 0.1i, c = 0.8i:



CCalculator.java

(c3a) Correctly implemented looping.
(c3b) Exit when unknown operation entered.
(c3c) Correct input of arguments.
(c3d) [up to 2*] Calls the right methods from Complex, does not repeat functionality.

(s3a) Outputs are formatted neatly.
(s3b) Menu formatted neatly.
(s3c) Input prompts formatted neatly.

Some useful test cases (as displayed by working code):

(1.0 + 2.0i) * (3.0 + 4.0i) = (-5.0 + 10.0i)

(2.0 + 4.0i) / (0.0 + 2.0i) = (2.0 - 1.0i)

|(3.0 - 4.0i)| = 5.0

ang((2.0 + 2.0i)) = 0.7853981633974483