CMPT 103 – Lab #10

General Information

Python version and IDE:	Python 3.3 / Wing IDE 101
Allocated lab time:	2 hrs and 50 min
Due date:	At the end of the lab period
Lab weight:	2%

Topics

- ✓ Recursion

Submission

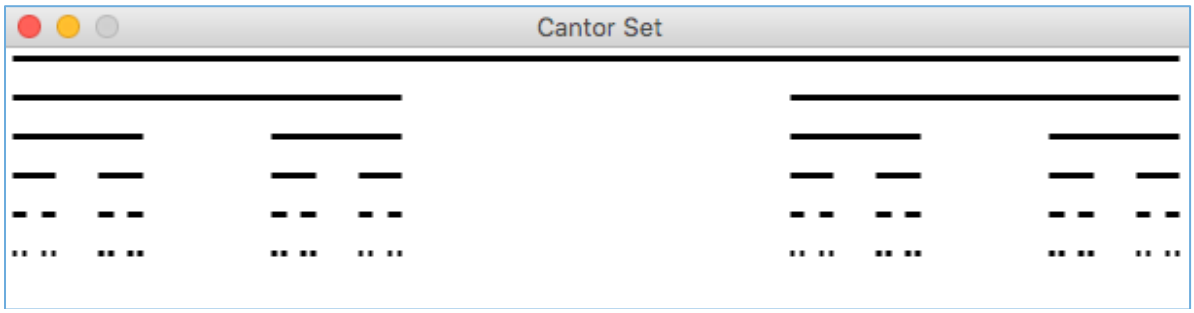
- ✓ **All code file (.py) should be submitted electronically** to your Lab Blackboard site.
- ✓ A portion of the total marks (20%) will be allocated for the programming style. For example, functions should be small; avoid writing duplicate code; names should be meaningful and descriptive; naming conventions should be followed consistently; code should be formatted properly; and comments should be accurate and well written.
- ✓ Comments are **required** for:
  - EACH program indicating the student name and program name.
  - EACH function indicating the function purpose, syntax (example usage of the function), parameters, and return value
  - Any block of code for which the purpose may be unclear (Note: you should always try to write clean code that can be understood easily without comments).

Assignment

For this lab, please put all functions into a file called `Lab10your_initials.py` (e.g., `Lab10FL.py` where F and L are the first letter of your first name and last name). Please feel free to write helper functions if necessary.

1. [30 marks] Write a recursive function called `sum_to` that accepts an integer `n` and returns the sum of the first `n` reciprocals. That is, `sum_to(n)` returns (1 + 1/2 + 1/3 + … + 1/n). For example, `sum_to(2)` should return 1.5. The function should return 0 if it is passed the value 0.

2. [35 marks] The Cantor set is a fractal that is defined by repeatedly removing the middle thirds of line segments as shown below.



Using graphics.py, write a recursive function called `cantor_set` that draws the Cantor set on a GraphWin object. Write another function to demonstrate that your program works properly. You should discuss your program design with your lab instructor.

3. [35 marks] Write a recursive function called `extract_unique_elements` that takes any nested list and returns a set containing unique values in the given list.

For example:

- `extract_unique_elements([2]) returns {2}`
- `extract_unique_elements([[[[5, 5]]]]) returns {5}`
- `extract_unique_elements([2, 2, 3, [[[3]]]]) returns {2, 3}`
- `extract_unique_elements([[3, [3, [[3]]]]]) returns {3}`
- `extract_unique_elements([2, [[2, 4, 5]], 'c', [[['c', 'd']]]]) returns {2, 4, 5, 'c', 'd'}`

Hint:

The `isinstance` function can help you solve this problem.
See https://docs.python.org/3.3/library/functions.html#isinstance

For example:

- `isinstance([5], list) returns True`
- `isinstance(5, list) returns False`