COMP130 Practice Lab Exam

Instructor: John MacCormick

Total points on exam: 115
Time allowed: 120 minutes

Create a new Python script called `practiceLabExam.py`. Write all answers in this script, and submit this script to Moodle before the end of the exam.

Qu 1. (10 points) Write a function with the signature `print_6_multiples(n)`. The function should print out the first six multiples of the parameter `n` on separate lines. Example: the output for `print_6_multiples(5)` should be

```
5
10
15
20
25
30
```

Qu 2. (10 points) Write a new function that generalizes your answer to the previous question by adding an extra parameter. The signature will be `print_multiples2(n, num_multiples)`. The function should print out the first `num_multiples` multiples of the parameter `n` on separate lines. Example: the output for `print_multiples(8, 3)` should be

```
8
16
24
```

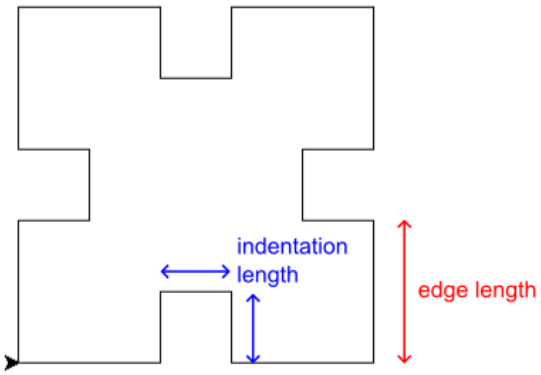
Qu 3. (10 points) Write a new function that is the same as in the previous question, but the sum of all multiples printed is also printed at the end. The signature will be `print_sum_of_multiples(n, num_multiples)`. Example: the output for `print_sum_of_multiples(7, 3)` should be

```
7
14
21
Sum is 42
```

Qu 4. (10 points) Write a new function that is the same as in Question 2, but only multiples that are even numbers are printed. The signature will be `print_even_multiples(n, num_multiples)`. Note that the parameter `num_multiples` represents the number of candidates that might be printed out, but only those that are even numbers will actually be printed. Example: the output for `print_even_multiples(9, 10)` should be

```
18
36
54
72
90
```

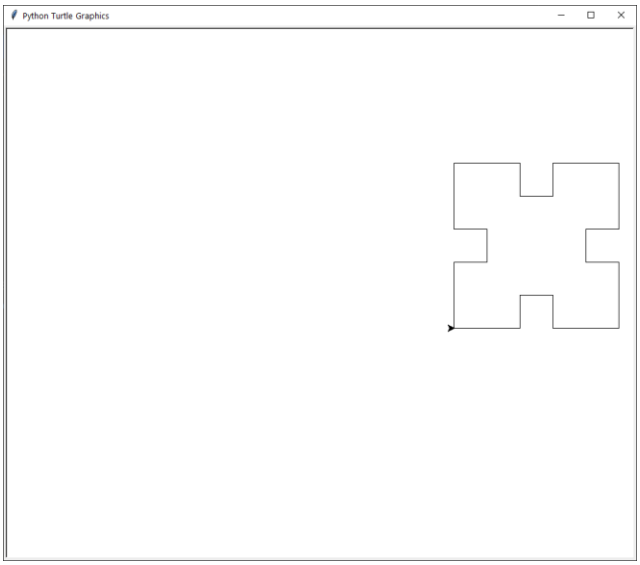
The remaining questions in this exam ask you to write functions using the turtle module. The questions will involve a shape that we call a *funny square*. A funny square is the same as a regular square except that it has small square-shaped indentations in each side, as shown on the right.



The *edge length* of a funny square is the number of pixels from one of the corners to the start of an adjacent indentation, and the *indentation length* is the number of pixels on each side of an indentation, as shown on the right.

The indentation length is always half of the edge length.

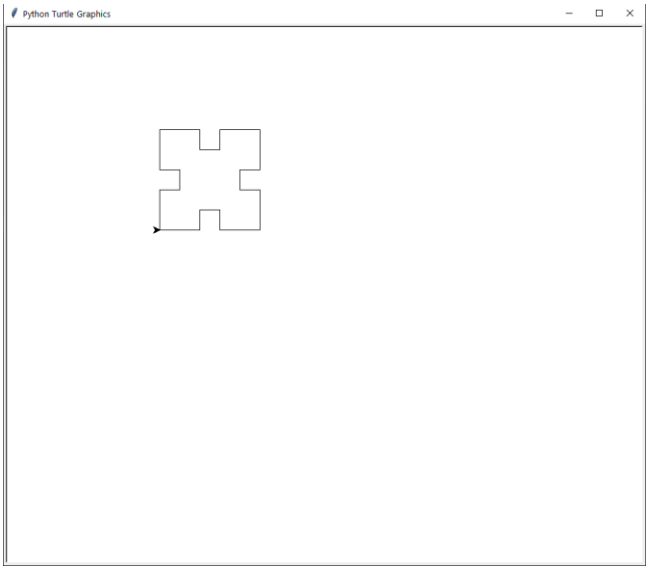
Qu 5. (20 points) Write a function with signature `funny_square1(t, x, y)` which draws a funny square whose bottom left corner is at position (`x`, `y`), with edge length 100 pixels. The first parameter, `t`, should be a Turtle object that is used for the drawing. Example: the output of `funny_square1(t, 200, -50)` should be as shown below.



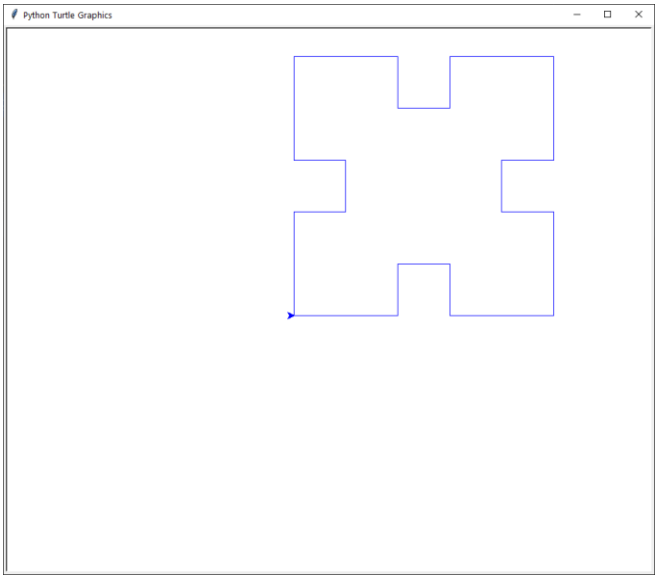
In the remaining questions, as in the previous question, the parameter `t` will always refer to a Turtle object used for the drawing.

Useful hint: For rapid experimentation, turn off turtle animation by executing the statement `turtle.tracer(0)` before drawing with the turtle. At the end of your program, use `turtle.update()` to ensure that everything is drawn to the screen.

Qu 6. (10 points) Write a new function that generalizes the function in the previous question. The new signature is `funny_square2(t, x, y, edge_len)`. The new parameter, `edge_len`, specifies the edge length of the funny square in pixels. Example: the output of `funny_square2(t, -250, 100, 60)` should be as shown below.

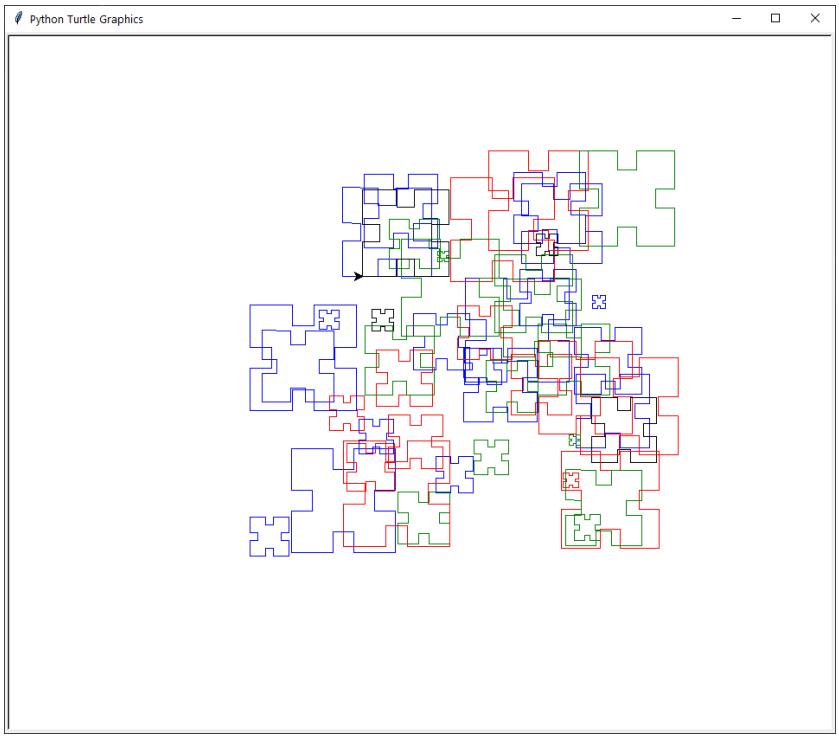


Qu 7. (20 points) Write a new function whose signature is `funny_square3(t, x, y, edge_len)`. The parameters have the same meaning as in the previous question, but now the color of the funny square can be different. The color is chosen according to the following rule. If `x` and `y` are both even, the color is red. If `x` and `y` are both odd, the color is green. Otherwise, the color is blue. However, if the edge length is a multiple of 10, then the color is black regardless of the `x` and `y` values. Example: the output of `funny_square3(t, -55, -20, 153)` should be as shown below.



Qu 8. (20 points) Write a new function with the signature `many_funny_squares(t, num_squares)`. This function will draw `num_squares` funny squares (in the appropriate colors as described in the previous question) at random locations and with random sizes. Specifically, for each funny square drawn: the x- and y-coordinates should be between -200 and 200 inclusive; and the edge length should be chosen randomly between 5 and 50 pixels inclusive.

Example: Here is a typical output for `many_funny_squares(t, 50)` -- although your output will differ due to randomness.



Note: The final question below is more challenging and time-consuming than the previous ones, but it is worth only a small amount of credit. Please do not attempt this question until you have completed all previous questions, tested them carefully, and submitted your practiceLabExam.py file to Moodle.

Qu 9. (5 points) Create a new Python script called `refactored.py`. You will submit this file to Moodle in addition to the `practiceLabExam.py` file. Copy all code from `practiceLabExam.py` into `refactored.py`. In the new script, factor out all repeated code from the functions for drawing funny squares. Use the software engineering techniques we have studied to refactor the code into high-quality source code that is reusable, maintainable, and extensible.