219116/117 Programming II

Midterm Examination

4 March 2021

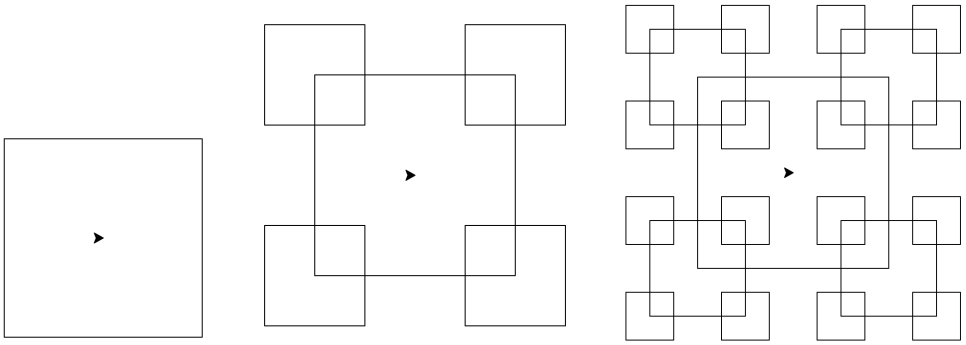
Task to do

1. (100) Complete the code in bank_account_template.py so that it generates a sample outcome similar to bank_accout_result.txt. Everyone result will be different because of the randomness during account generation. **Do not modify other parts of the code except where you need to add in your code marked `# fill in your code here.`** Note that:

- A checking account interest rate is 0.01; a saving account interest rate is 0.02
- A checking account charges 2 unit for every withdraw
- A saving account charges 1 unit for every deposit

Once you are done, save the completed version of your work to a file named bank_account.py

2. (100) Produce a recursive square drawing of order 4 given that drawings for order 1, 2, and 3 have the following patterns.



Order 1 Order 2 Order 3

Use the code in recursive_squares_template.py as a guide. Fill in the missing part and save the completed version in a file named recursive_squares.py

3. (50) The following is an example of a repeatable function. It returns another repeatable function in a mutual recursive manner.

```
def growth(baseline):
    """Return a function that can be called repeatedly on numbers and
prints the difference between its argument and the smallest argument used
so far (including baseline).

    >>> job = growth(148)(149)(150)(130)(133)(139)(137)
    1
    2
    0
    3
    9
    7
    """
    def increase(observed):
        under = min(baseline, observed)
```

```
        print(observed - under)
        return growth(under)
    return increase
```

The function `growth` takes a number `baseline` and returns a repeatable function `increase`. When `increase` is called on a number `observed`, it prints the difference between `observed` and the smallest argument passed to `growth` or `increase` so far among the repeated calls.

Once you understand the code above, fill in the missing code in HOF_maxer_template.py so that it passes all the test cases provided in the Doctest for the maxer function. Save the completed code as HOF_maxer.py

4. (50) Write a report in a file named midterm_report.pdf. In the report,
   - Explain if you complete each problem 100% or you encounter some bugs; describe the nature of the bugs
   - Attach screenshots of your source code and your run results side-by-side

A good writeup of the report will earn you up to 50 points.

No report = zero midterm score

Submission:

- *Create StudentID_Firstname_midterm folder, where StudentID is your KU ID and Firstname is your given name*
- *Put the files to submit, `bank_account.py`, `recursive_squares.py`, `HOF_maxer.py`, into this folder along with the mandatory `midterm_report.pdf`*
- *Zip the folder and submit the zip file to the course's Google Classroom at the end of the exam*