MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

CENG 111

Fall 2024

Take-Home Exam 3

REGULATIONS

Due date: 23:59, 25 December 2024, Wednesday *(Not subject to postpone)*

Submission: Electronically. You should save your program source code as a text file named `the3.py`. Check the announcement on the ODTUCLASS course page for the submission procedure.

Team: There is **no** teaming up. This is an EXAM.

Cheating: Source(s) and Receiver(s) will receive zero and be subject to disciplinary action.

INTRODUCTION

In this take-home exam, your task is to count the number of rectangles in a binary 2D pattern, P, with dimensions $H \times W$. The pattern, being binary, consists of only 0s and 1s. The rectangles are *empty*, containing 0s. However, rectangles can overlap with each other or include other rectangles in themselves. The overlaps among rectangles can introduce new rectangles, which should be counted as well.

Figure 1 provides an example pattern with the following four rectangles (a rectangle is specified with its corners as [(top-left-row, top-left-column), (bottom-right-row, bottom-right-column)]):

- Red rectangle: [(2,2), (5,7)].
- Green rectangle: [(4,6), (6,9)].
- Blue rectangle: [(4,4), (9,9)].
- Orange rectangle: [(4,6), (6,9)].

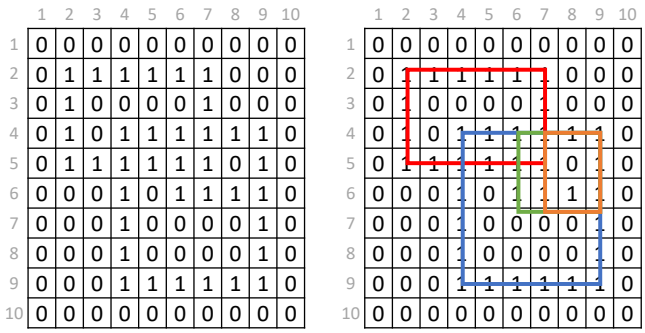

Figure 1: An example patter (left) and the rectangles to be counted (right).

PROBLEM & SPECIFICATIONS

- You should implement a function `count_rectangles(Pattern)` that takes a single parameter and returns the count of rectangles (`int`).
- A `Pattern` is going to be represented as a list of strings where each row is represented by a string. In other words, each string in the list has the same length.
- For example, the pattern in Figure 1 can be represented as follows:

```
[
"0000000000",
"0111111000",
"0100001000",
"0101111110",
"0111111010",
"0001011110",
"0001000010",
"0001000010",
"0001111110",
"0000000000",
]
```

- Rectangles are going to be parallel to the axes of the coordinate system.
- For a rectangle to be valid, its borders should be marked with 1s and it should contain at least one 0 (zero).
- It is guaranteed that the pattern P will not contain non-rectangular shapes and that any 1 (one) in the pattern is guaranteed to be part of a rectangle.

SAMPLE RUN

```
>>> P = [
"0000000000",
"0111111000",
"0100001000",
"0101111110",
"0111111010",
"0001011110",
"0001000010",
"0001000010",
"0001111110",
"0000000000",
]
>>> count_rectangles(P)
4
```

RESTRICTIONS & GRADING

- You are free to use recursion or iteration, and define as many functions as necessary.

- You are not allowed to import any libraries.
- A set of arbitrarily selected students will be subject to oral examination about their solutions. The details will be announced on ODTUclass later.
- Your program will be graded through an automated process and therefore, any violation of the specifications will lead to errors (and reduction of points) in automated evaluation. You should especially avoid printing something on screen.
- Your solutions will not be tested with incorrect or erroneous inputs.
- Your program will be tested with multiple data (a distinct run for each data). Any program that performs only 30% and below will enter a glass-box test (eye inspection by the grader TA). The TA will judge an overall grade in the range of [0,30].
- A program based on randomness will be graded zero.
- The glass-box test grade is neither open to discussion nor explanation.