CMPE 326 Concepts of Programming Languages

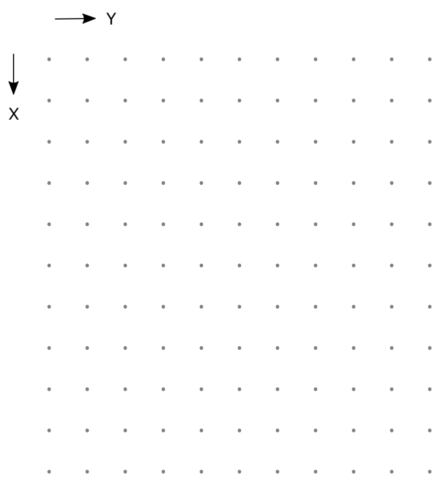
Spring 2021

Homework 1

Due date: 01/04/2021 23:59

In this homework you will develop a system in Python that helps controlling a laser logo printer machine (you can also see it as a part of 3D printer). The machine can be used to engrave some figures (logos) on a metal or wood surface. The system controls the machine's laser head.

The surface has a grid layout and its size is fized. There are 11x11 dots. Below is an empty surface with the grid dots.



The laser head can move one grid unit up, down, left and right. By controlling these movements the machine can actually engrave a logo on the surface.

Your system must implement 3 main user commands. It must accept commands from the **standard input** and write all the output to the **standard output**.

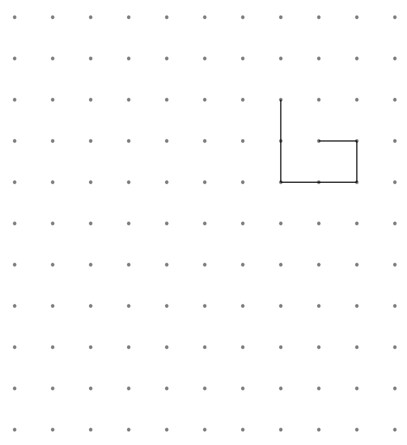
LOGO logo1 DDRRUL

The `LOGO` command tells the system what movements of the laser head comprises a logo. After keyword `LOGO` the user needs to enter a name of the logo as a string with no spaces (in the above example it is `logo1`) and then the ordered movements of the laser head. To this end, characters `D`, `U`, `L`, and `R` are used for down, up, left, and right movements, respectively. For the above example, the head moves down, down, right, right, up and then left. The user may enter as many logo definitions as needed using the `LOGO` command. You can assume that the user does not use any logo name twice. The above `LOGO` command outputs the following message.

```
logo1 defined
```

ENGRAVE logo1 3 8

The `ENGRAVE` command controls the laser head to engrave a logo. It expects first the name of the logo. This logo must be defined earlier via `LOGO` command. Next, it expects the x and y-coordinates of the grid dot where the laser head should start engraving. While the first number is the x-coordinate that designates the row of the grid dot, the second one is the y-coordinate designating the column of the dot. The top left-most grid dot is (1,1) and the bottom right-most one is (11,11). For example, the above command engraves the logo `logo1` starting from the (3,8) dot on the surface. After the `ENGRAVE` command, the system must output the textual representation of the grid surface that shows the engraved logo. Below are the engraved logo `logo1` and its textual representation as the output of above command. Note that in the output, you must use the `-` character for horizontal engraved lines, `|` character for vertical engraved lines, and `.` character for grid dots. The symbol ␣ is used to designate the space character.



```
.␣.␣.␣.␣.␣.␣.␣.␣.␣.␣.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
.␣.␣.␣.␣.␣.␣.␣.␣.␣.␣.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
.␣.␣.␣.␣.␣.␣.␣.␣.␣.␣.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣␣␣␣␣␣
.␣.␣.␣.␣.␣.␣.␣.␣.-.␣.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣␣␣|␣␣
.␣.␣.␣.␣.␣.␣.␣.-.-.␣.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
.␣.␣.␣.␣.␣.␣.␣.␣.␣.␣.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
.␣.␣.␣.␣.␣.␣.␣.␣.␣.␣.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
.␣.␣.␣.␣.␣.␣.␣.␣.␣.␣.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
.␣.␣.␣.␣.␣.␣.␣.␣.␣.␣.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
.␣.␣.␣.␣.␣.␣.␣.␣.␣.␣.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
.␣.␣.␣.␣.␣.␣.␣.␣.␣.␣.
```

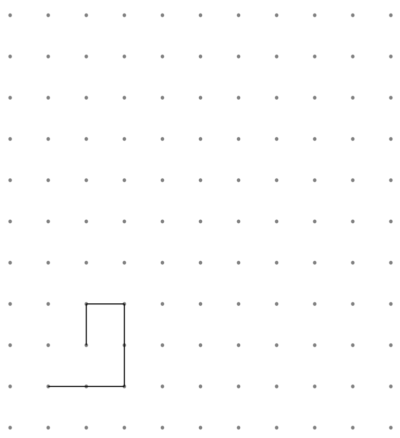
SAME logo1 logo2

The `SAME` command checks whether two logos have the same shape irrespective of their orientation. It expects the names of two logos that have been defined earlier via `LOGO` command. When comparing two logos we are just considering their shapes neither their absolute positions on the surface nor their orientation, i.e., we may turn a logo engraved surface 90°, 180°, and 270° clockwise (or counter-clockwise) when comparing with another logo. The command outputs `Yes` if the two logos match or `No` otherwise.

Consider the following two commands that define `logo2` and engrave it.

```
LOGO logo2 URDUDDLL
logo2 defined
ENGRAVE logo2 10 3
```

The following figure depicts the logo logo2.



Note that if we turn the surface of logo2 90° clockwise or turn the surface of logo1 90° counter-clockwise, both logos match. Hence, the following `SAME` command outputs `Yes`.

```
SAME logo1 logo2
Yes
```

Implementation Language

You must use Python as the implementation language. You can appreciate some of the properties of Python while performing this task.

First of all since Python is dynamically typed language, you can prototype and develop programs fast.

Additionally, high-level data structures like *lists* and *dictionaries* are built-in for Python. Hence, you can benefit from them to form advanced data structures easily to be used in this homework. Note that these collection data structures can easily hold elements of different types. For instance, `[('logo', 2, 4), ('logo2', 3, 4), 3, 'CMPE326']` is a valid Python list. With this feature, you can build up complex structures.

Note that Python features the closure concept that you may benefit in this homework. Using closures you can write functions that return functions. These returned functions can be assigned to variables and later be called via respective variables. You may find closures handy when implementing logo definitions.

You will be provided with some test cases by your teaching assistant. You must follow the output specifications strictly. Otherwise (even a single space character difference between your output and expected output), you will loose points.

Submission

Each person must submit **his or her own work**.

Information about submitting your homework via Moodle will be provided by your teaching assistant.

You are **not allowed** to use special packages. You can only use modules from the standard Python libraries.

Your program will be evaluated using the Python 3.X (min version 3.6) interpreter.

You may be asked for a demo session.

The final submission must be one file named `hw1.py`.

There will be 2 days **questions fence** for this homework. You are not allowed to ask questions to the instructor or the teaching assistant in 2 days period before the deadline.

Your submission will be graded w.r.t. the maximum points calculated according to the following formula: $100 - (2^{NumOfLateDays} \times 5)$.