Experiments

The folder **"CS_Task2_Experiments"** contains a folder **"Test_Images"** which comprises 5 test images (test_image1.jpg, test_image2.jpg, …, test_image5.jpg). These 5 test images will serve as 5 **test cases**.

Teams need to write the code for two experiments by modifying **"CS_Task2_Experiments.py".** The code should satisfy all the test cases for the two experiments listed below.

Experiment 1: Converting Grid to Arrays **(15 Marks)**

Each test image represents a **5*5 grid**. The **grid** is made up of **nodes** that can be represented by **X and Y Co-ordinates (X, Y),** each numbered from **0 to 5** as shown in Figure 1.

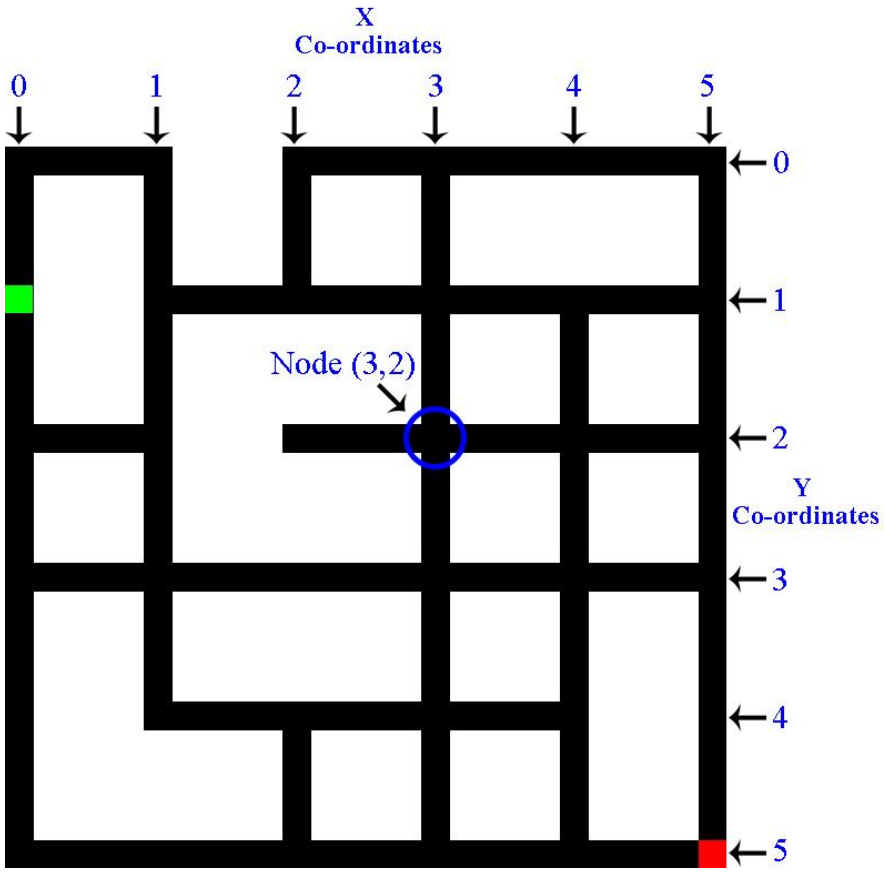

Figure 1: A test image with co-ordinates

For example, the node highlighted by the blue circle can be represented as (X, Y) = (3, 2) as shown in Figure 1.

A black line connecting any two nodes is called a **link.** The grid is made up of horizontal and vertical links that connect the nodes; some links may be missing. Please refer to Figure 2 for types of links.

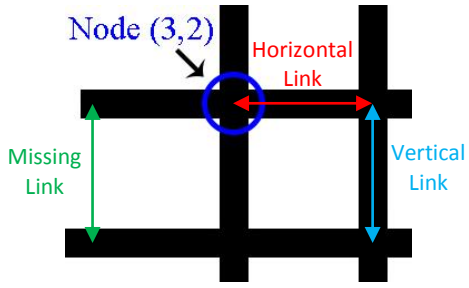

Figure 2: Types of links

Problem Statement:

Write a function in Python to convert the grid into **two arrays**. A **Horizontal Links array** (6*5 array) and a **Vertical Links array** (5*6 array).

For the Horizontal Links array, you need to check for Horizontal Links between two neighboring nodes in each **row**. If a Horizontal Link is present we denote it by **1.** If a Horizontal Link is missing we denote it by **0.** For example, for Figure 1, the Horizontal Links array should be:

Horizontal Links array:
[[1, 0, 1, 1, 1],
[0, 1, 1, 1, 1],
[1, 0, 1, 1, 1],
[1, 1, 1, 1, 1],
[0, 1, 1, 1, 0],
[1, 1, 1, 1, 1]]

Similarly, for the Vertical Links array, you need to check for Vertical Links between two neighboring nodes in each **column**. If a Vertical Link is present we denote it by **1.** If a Vertical Link is missing we denote it by **0.** For example, for Figure 1, the Vertical Links array should be:

Vertical Links array:
[[1, 1, 1, 1, 0, 1],
[1, 1, 0, 1, 1, 1],
[1, 1, 0, 1, 1, 1],
[1, 1, 0, 1, 1, 1],
[1, 0, 1, 1, 1, 1]]

You are required to modify a function **grid_to_arrays(img)** which takes an image as input and returns two lists – horizontal_links and vertical_links.

Open **"CS_Task2_Experiments.py"** in Python IDLE editor and add your code to the following code snippet. Inline comments are mandatory to explain the code.

```
def grid_to_arrays(img):
    '''
    * Function Name: grid_to_arrays
    * Input: img - Any one of the test images
    * Output: arrays -- a list containing the Horizontal Links array and the
                        Vertical Links array
    * Example Call: a, b = grid_to_arrays(img)
                    >>> a = horizontal_links, b = vertical_links
    '''
    #add your code here
    return horizontal_links, vertical_links
```

Experiment 2: Path Planning **(25 Marks)**

Each test image contains:

- A **green marker** denoting the **Start.**
- A **red marker** denoting the **End.**

The objective of this experiment is to find the **shortest path** from Start to End.

You need to start from the Start and reach the End by moving either horizontally or vertically. The **length** of the path is determined by the number of nodes travelled (excluding the Start node).

You are required to modify a function **shortest_path(img)** which takes an image as input and returns the length and coordinates of a shortest path.

Note: In case of more than one shortest path, you may return any one of them.

For example, consider Figure 3:

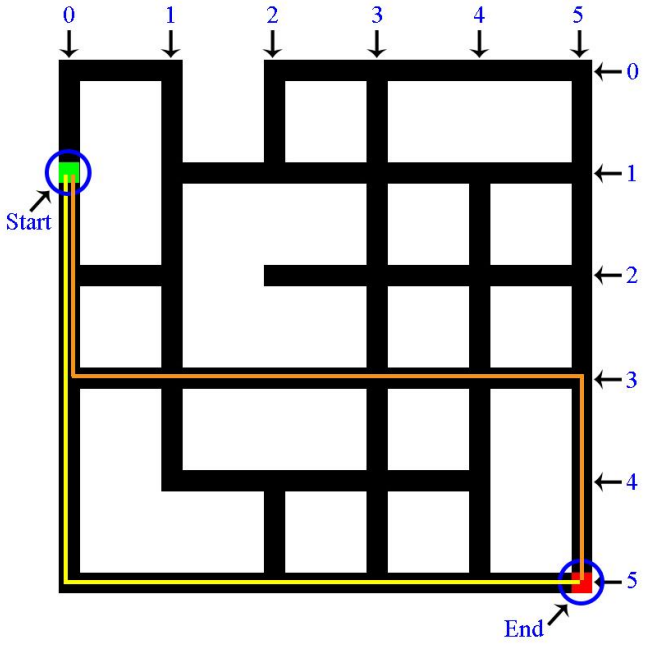

Figure 3: A test image

Start = **(0, 1)**
End = **(5, 5)**

The shortest path can be:
[(0, 2), (0, 3), (0, 4), (0, 5), (1, 5), (2, 5), (3, 5), (4, 5), (5, 5)]
OR
[(0, 2), (0, 3), (1, 3), (2, 3), (3, 3), (4, 3), (5, 3), (5, 4), (5, 5)]
OR …

Open **"CS_Task2_Experiments.py"** in Python IDLE editor and add your code to the following code snippet. Inline comments are mandatory to explain the code.

```
def shortest_path(img):
    '''
    * Function Name: shortest_path
    * Input: img - Any one of the test images
    * Output: length - length of shortest path
              shortest_path - the shortest path as a list of coordinates of
                              form (x,y)
    * Example Call: l, sp = shortest_path(img)
                    >>> l = length, sp = shortest_path
    '''
    #add your code here
    return length, shortest_path
```