10. Destination City You are given the array paths, where paths[i] = [cityAi, cityBi] means there exists a direct path going from cityAi to cityBi. Return the destination city, that is, the city without any path outgoing to another city. It is guaranteed that the graph of paths forms a line without any loop, therefore, there will be exactly one destination city. Example 1: Input: paths = [["London","New York"],["New York","Lima"],["Lima","Sao Paulo"]] Output: "Sao Paulo" Explanation: Starting at "London" city you will reach "Sao Paulo" city which is the destination city. Your trip consist of: "London" -> "New York" -> "Lima" -> "Sao Paulo". Example 2: Input: paths = [["B","C"],["D","B"],["C","A"]] Output: "A" Explanation: All possible trips are: "D" -> "B" -> "C" -> "A". "B" -> "C" -> "A". "C" -> "A". "A". Clearly the destination city is "A". Example 3: Input: paths = [["A","Z"]] Output: "Z" Constraints: 1 <= paths.length <= 100 paths[i].length == 2 1 <= cityAi.length, cityBi.length <= 10 cityAi != cityBi All strings consist of lowercase and uppercase English letters and the space character

PROGRAM:-

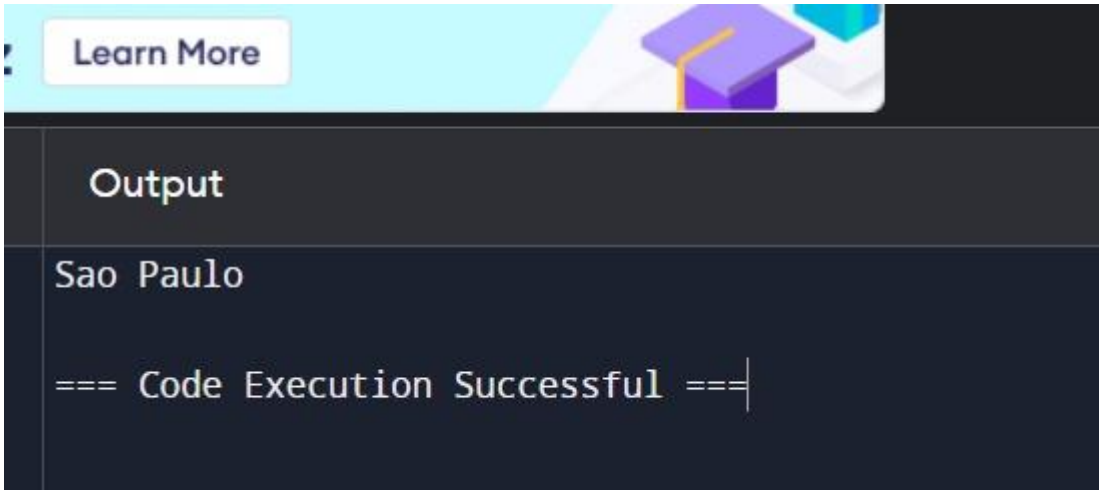
```
def destCity(paths):
    start_cities = set()
    end_cities = set()

    for path in paths:
        start_cities.add(path[0])
        end_cities.add(path[1])

    return (end_cities - start_cities).pop()

# Example
paths = [["London", "New York"], ["New York", "Lima"], ["Lima", "Sao Paulo"]]
print(destCity(paths))  # Output: "Sao Paulo"
```

OUTPUT:-



TIME COMPLEXITY:-O(n)