Shri Vile Parle Kelavani Mandal's

INSTITUTE OF TECHNOLOGY

DHULE (M.S.)

DEPARMENT OF COMPUTER ENGINEERING

Subject : Competitive Programmimg Lab

Name : Jaykishan Natwar Varma **Roll No. :** 68

Class : TY. Comp. Engg. **Batch :** T4 **Division:**

Expt. No. : **Date :**

Title : Shell Sort

Remark

Signature

ASSIGNMENT/EXPERIMENT: ___________________________		
Date of Performance:	**Date of Submission:**	
Marks Split Up	**Maximum Marks**	**Marks Obtained**
Performance/Conduction	**3**	
Report Writing	**3**	
Attendance	**2**	
Viva/Oral	**2**	
Total Marks	**10**	
Signature of Subject Teacher		

Title: Shell Sort

Aim: *He made each turtle stand on another one's back*
And he piled them all up in a nine-turtle stack.
And then Yertle climbed up. He sat down on the pile.
What a wonderful view! He could see 'most a mile!

King Yertle wishes to rearrange his turtle throne to place his highest-ranking nobles and closest advisors nearer to the top. A single operation is available to change the order of the turtles in the stack: a turtle can crawl out of its position in the stack and climb up over the other turtles to sit on
the top.
Given an original ordering of a turtle stack and a required ordering for the same turtle stack, your job is to determine a minimal sequence of operations that rearranges the original stack into the required
stack.

Language used: Python

Platform Used: Pycharm, VS code etc.

Sample Input: The first line of the input consists of a single integer *K* giving the number of test cases. Each test case consist on an integer *n* giving the number of turtles in the stack. The next *n* lines specify the original ordering of the turtle stack. Each of the lines contains the name of a turtle, starting with the turtle on the top of the stack and working down to the turtle at the bottom of the stack. Turtles have unique names, each of which is a string of no more than eighty characters drawn from a character set consisting of the alphanumeric characters, the space character and the period (`.'). The next *n* lines in the input gives the desired ordering of the stack, once again by naming turtles from top to bottom. Each test case consists of exactly 2*n* + 1 lines in total. The number of turtles (*n*) will be less than or equal to two hundred.

Sample Output: For each test case, the output consists of a sequence of turtle names, one per line, indicating the order in which turtles are to leave their positions in the stack and crawl to the top. This sequence of operations should transform the original stack into the required stack and should be as short as possible. If more than one solution of shortest length is possible, any of the solutions may be reported. Print a blank line after each test case.

Example:

Sample Input :

```
2
3
Yertle
Duke of Earl
Sir Lancelot
Duke of Earl
Yertle
```

```
Sir Lancelot
9
Yertle
Duke of Earl
Sir Lancelot
Elizabeth Windsor
Michael Eisner
Richard M. Nixon
Mr. Rogers
Ford Perfect
Mack
Yertle
Richard M. Nixon
Sir Lancelot
Duke of Earl
Elizabeth Windsor
Michael Eisner
Mr. Rogers
Ford Perfect
Mack
```

Sample Output:

```
Duke of Earl

Sir Lancelot
Richard M. Nixon
Yertle
```

Algorithm/Flowchart:

1. **Input Reading**:
    - Read the number of test cases `K`.
    - For each test case:
        - Read the number of turtles `n`.
        - Read the initial ordering of turtles into a list `initial_order`.
        - Read the desired ordering of turtles into a list `desired_order`.
2. **Mapping Turtles**:
    - Create a mapping (dictionary) `position_map` where the key is the turtle's name and the value is its current position in the `initial_order`.

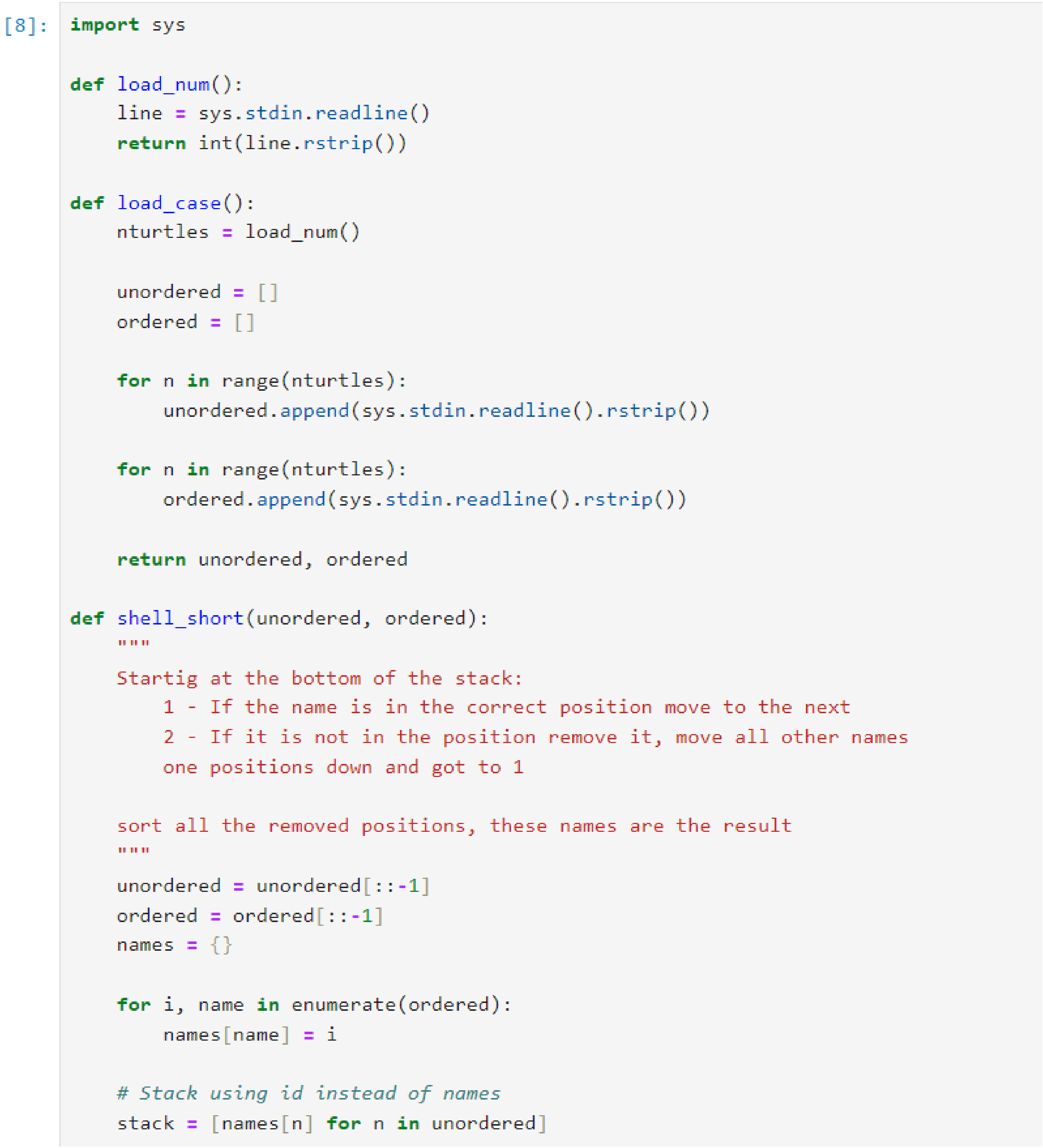
3. **Finding Moves**:

    - For each turtle in the `desired_order`, determine its current position in the `initial_order` using `position_map`.
    - If the turtle is not already at the top of the stack (i.e., its current position is not the top of `initial_order`), calculate the necessary moves to bring it to the top:
        - Simulate moving the turtle to the top by iteratively swapping it upwards until it reaches the desired position.

4. **Output**:

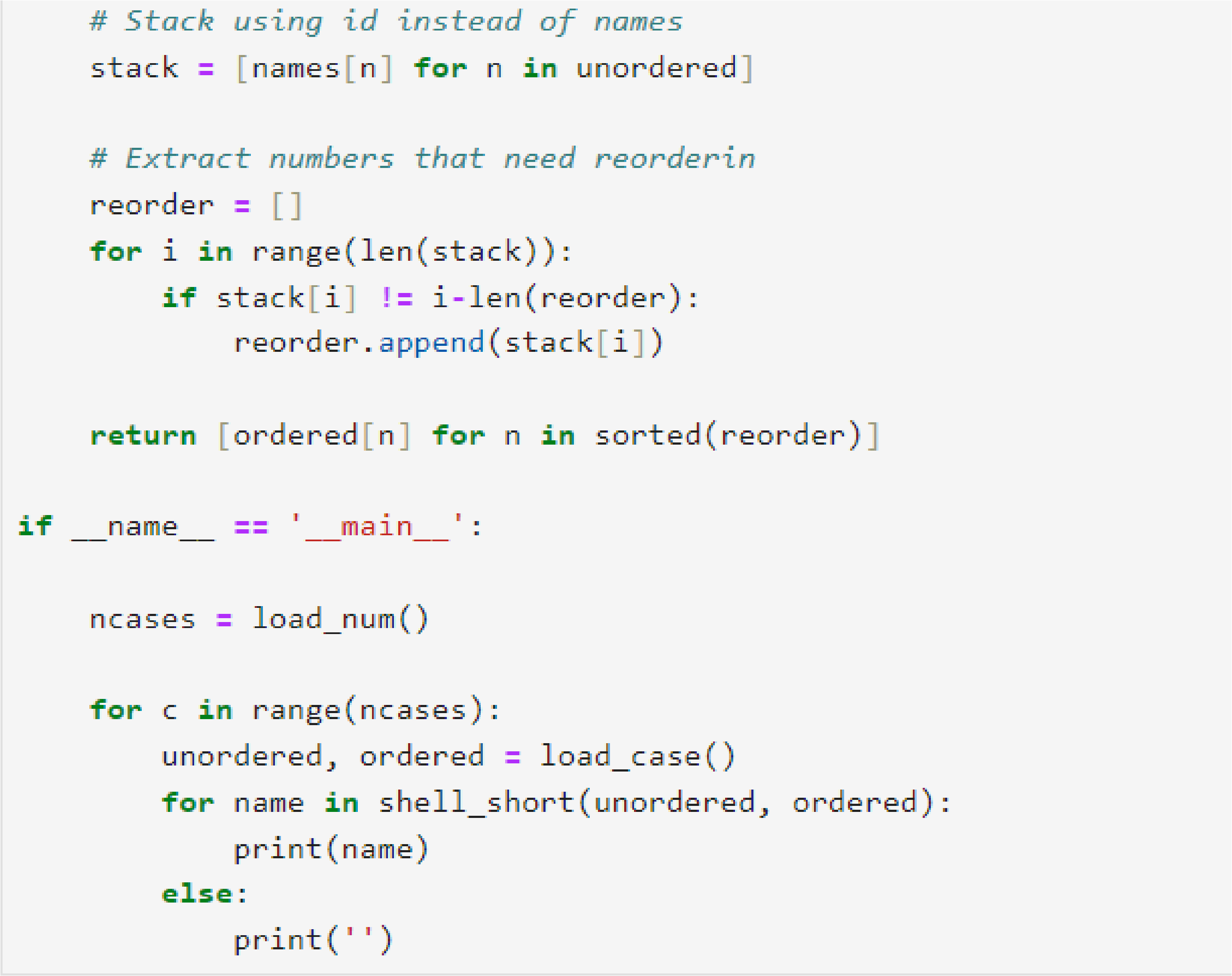
    - Print each move (turtle name) required to transform the `initial_order` into the `desired_order` for each test case.
    - Print a blank line after each test case.

Code:-



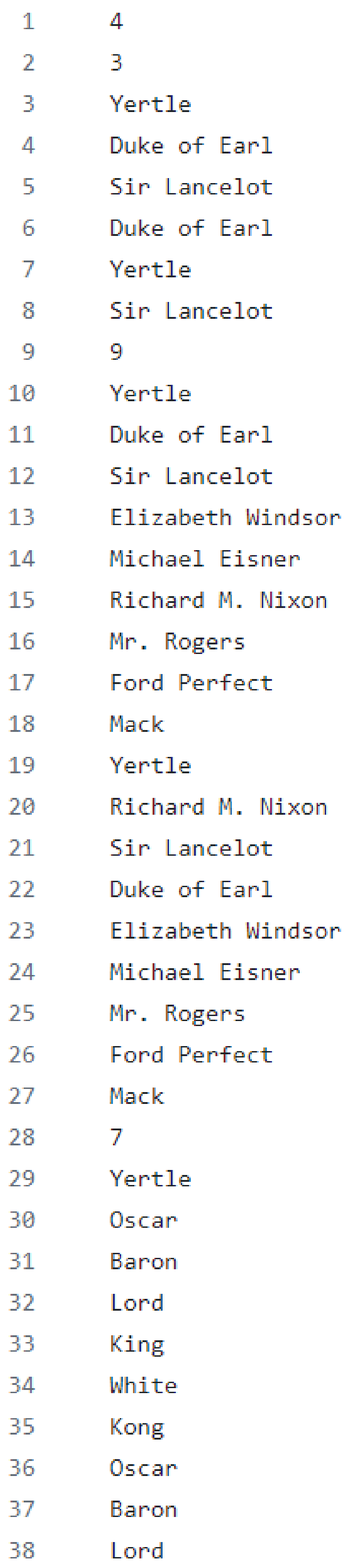
[8]:

```
import sys

def load_num():
    line = sys.stdin.readline()
    return int(line.rstrip())

def load_case():
    nturtles = load_num()

    unordered = []
    ordered = []

    for n in range(nturtles):
        unordered.append(sys.stdin.readline().rstrip())

    for n in range(nturtles):
        ordered.append(sys.stdin.readline().rstrip())

    return unordered, ordered

def shell_short(unordered, ordered):
    """
    Startig at the bottom of the stack:
        1 - If the name is in the correct position move to the next
        2 - If it is not in the position remove it, move all other names
        one positions down and got to 1

    sort all the removed positions, these names are the result
    """
    unordered = unordered[::-1]
    ordered = ordered[::-1]
    names = {}

    for i, name in enumerate(ordered):
        names[name] = i

    # Stack using id instead of names
    stack = [names[n] for n in unordered]
```

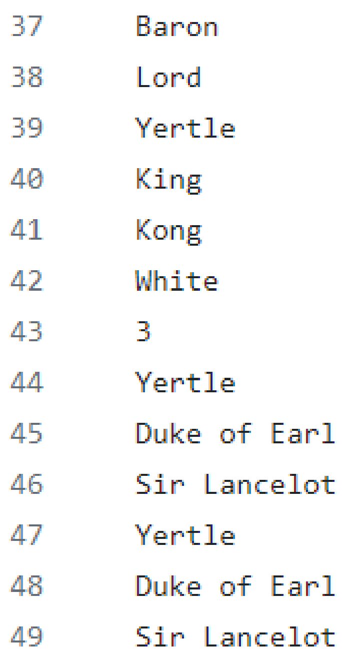


```python
    # Stack using id instead of names
    stack = [names[n] for n in unordered]

    # Extract numbers that need reorderin
    reorder = []
    for i in range(len(stack)):
        if stack[i] != i-len(reorder):
            reorder.append(stack[i])

    return [ordered[n] for n in sorted(reorder)]

if __name__ == '__main__':

    ncases = load_num()

    for c in range(ncases):
        unordered, ordered = load_case()
        for name in shell_short(unordered, ordered):
            print(name)
        else:
            print('')
```

Input:-

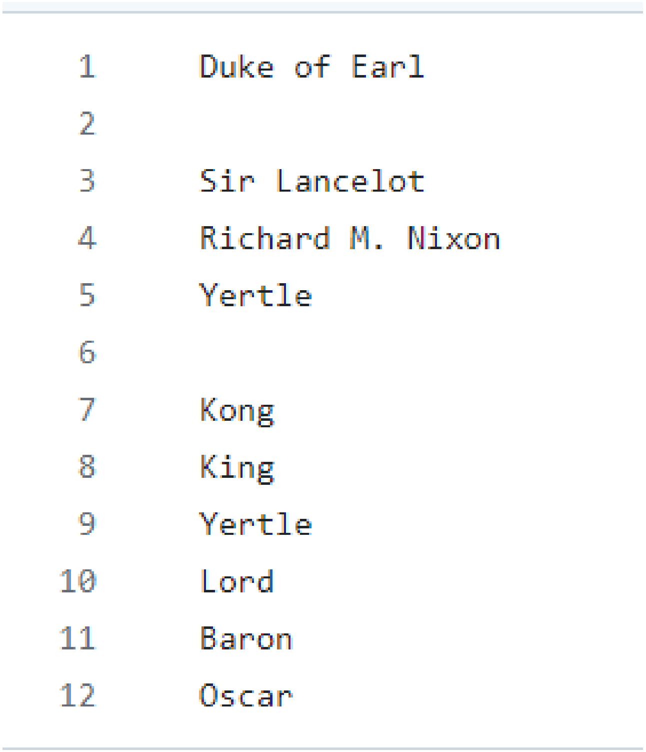


```
4
3
Yertle
Duke of Earl
Sir Lancelot
Duke of Earl
Yertle
Sir Lancelot
9
Yertle
Duke of Earl
Sir Lancelot
Elizabeth Windsor
Michael Eisner
Richard M. Nixon
Mr. Rogers
Ford Perfect
Mack
Yertle
Richard M. Nixon
Sir Lancelot
Duke of Earl
Elizabeth Windsor
Michael Eisner
Mr. Rogers
Ford Perfect
Mack
7
Yertle
Oscar
Baron
Lord
King
White
Kong
Oscar
Baron
Lord
```



```
Baron
Lord
Yertle
King
Kong
White
3
Yertle
Duke of Earl
Sir Lancelot
Yertle
Duke of Earl
Sir Lancelot
```

Output:-



```
Duke of Earl

Sir Lancelot
Richard M. Nixon
Yertle

Kong
King
Yertle
Lord
Baron
Oscar
```

Conclusion: In this way we implement The Shell Sort Problem using loops and conditional statements.