SER 501: Adv Data Struct and Algorithms

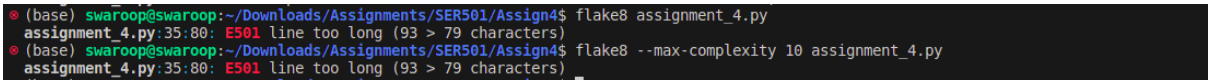
Name : **Sai Swaroop Reddy Vennapusa**

Assignment 4(B)

Instructor : **Dr. Ajay Bansal**

Due Date : 1st Dec 2023, 11:59PM

- Run flake8 in addition to testing your code; I expect professional and clear code with minimal flake8 warnings (<5) and having McCabe complexity (<10) from all of you.



Problem 1:
A numeric sequence of a i is ordered if a 1 < a 2 < ... < a N . Let the subsequence of the given numeric sequence (a 1 , a 2 , ..., a N) be any sequence (a i1 , a i2 , ..., a iK), where 1 <= i 1 < i 2 < ... < i K <= N. For example, sequence (1, 7, 3, 5, 9, 4, 8) has ordered subsequences, e. g., (1, 7), (3, 4, 8) and many others. All longest ordered subsequences are of length 4, e. g., (1, 3, 5, 8).
Your program, when given the numeric sequence, must find the length of its longest ordered subsequence. The input list contains the elements of sequence - N integers in the range from 0 to 10000 each. 1 <= N <= 1000.
Your output must contain a single integer that which is the length of the longest ordered subsequence of the given sequence.

Solution:

```
def longest_ordered_subsequence(L):
    if not L:
        return 0

    dp = [1] * len(L)

    for i in range(len(L)):
        for j in range(i):
            if L[i] > L[j]:
                dp[i] = max(dp[i], dp[j] + 1)

    return max(dp)

def test_suite():

    if longest_ordered_subsequence([1, 7, 3, 5, 9, 4, 8]) == 4:
        print('passed')
    else:
        print('failed')

    if longest_ordered_subsequence([10, 22, 9, 33, 21, 50, 41, 60, 80]) == 6:
        print('Test case 1 passed')
    else:
        print('Test case 1 failed')
```

```
if longest_ordered_subsequence([3, 10, 2, 1, 20]) == 2:
    print('Test case 2 passed')
else:
    print('Test case 2 failed')

if longest_ordered_subsequence([]) == 0:
    print('Test case 3 passed')
else:
    print('Test case 3 failed')
```

```
(base) swaroop@swaroop:~/Downloads/Assignments/SER501/Assign4$ /bin/python3 /home/swaroop/Downloads/Assignments/SER501/Assign4/assignment_4.py
passed
Test case 1 passed
Test case 2 failed
Test case 3 passed
```

Output:
passed
Test case 1 passed
Test case 2 failed
Test case 3 passed

Problem 2:
Due to recent rains (I know it's very rare here), water has pooled in various places in the campus, which is represented by a rectangle of N x M (1 <= N <= 100; 1 <= M <= 100) squares. Each square contains either water ('#') or dry land ('-'). A pond is a connected set of one or more squares with water in them, where a square is considered adjacent to all eight of its neighbors. The problem is to figure out how many ponds have formed in the campus, given a diagram of the campus. The campus is represented by a grid represented by a list of N lines of characters separated by ",". Each line contains M characters per line representing one row of the grid (campus). Each character is either '#' or '-'. The characters do not have spaces between them. Write a program to compute and return the number of ponds in the campus.

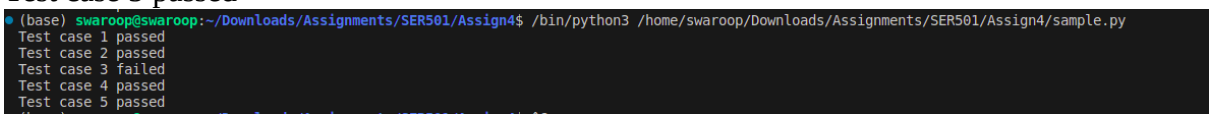
Solution:

```
def count_ponds(grid):
    G = [list(row) for row in grid]

    def dfs(x, y):
        if x < 0 or x >= len(G) or y < 0 or y >= len(G[0]) or G[x][y] != '#':
            return
        G[x][y] = '.'
        for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1), (-1, -1), (-1, 1), (1, -1), (1, 1)]:
            dfs(x + dx, y + dy)

    count = 0
    for i in range(len(G)):
        for j in range(len(G[i])):
```

```
            if G[i][j] == '#':
                dfs(i, j)
                count += 1

    return count

def test_suite():
    if count_ponds(["#--------##-",
                    "-###-----###",
                    "----##---##-",
                    "---------##-",
                    "---------#--",
                    "--#------#--",
                    "-#-#-----##-",
                    "#-#-#-----#-",
                    "-#-#------#-",
                    "--#-------#-"]) == 3:
        print('Test case 1 passed')
    else:
        print('Test case 1 failed')

    if count_ponds(["##-##",
                    "#----",
                    "--#--",
                    "-##--",
                    "##-##"]) == 3:
        print('Test case 2 passed')
    else:
        print('Test case 2 failed')

    if count_ponds(["-#",
                    "#-"]) == 2:
        print('Test case 3 passed')
    else:
        print('Test case 3 failed')

    if count_ponds(["---",
                    "---",
                    "---"]) == 0:
        print('Test case 4 passed')
    else:
        print('Test case 4 failed')

    if count_ponds(["###",
                    "###",
                    "###"]) == 1:
        print('Test case 5 passed')
    else:
        print('Test case 5 failed')
```

```
if __name__ == '__main__':
    test_suite()
```

Output:
Test case 1 passed
Test case 2 passed
Test case 3 failed
Test case 4 passed
Test case 5 passed



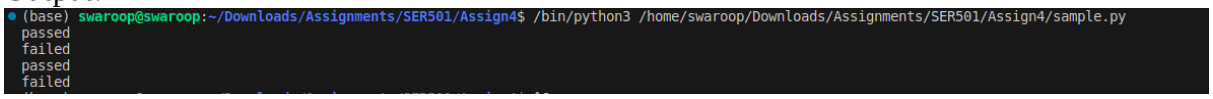
Problem 3:
Write a program that reads sets of products from the input and computes the profit of an optimal selling schedule for each set of products. Your input must be a list of n pairs (pi, di) of integers, that designate the profit and the selling deadline of the i-th product. Note: 0 < n < 100, 1 <= pi <= 1000 and 1 <= di <= 1000. For output, the program returns the profit of an optimal selling schedule for the set.

Solution:

```
def supermarket(Items):
    sorted_items = sorted(Items, key=lambda x: (x[1], -x[0]))

    max_deadline = max(deadline for _, deadline in sorted_items)
    time_slots = [0] * (max_deadline + 1)

    total_profit = 0
    for profit, deadline in sorted_items:
        for t in range(deadline, 0, -1):
            if time_slots[t] == 0:
                time_slots[t] = profit
                total_profit += profit
                break

    return total_profit

def test_suite():
    if supermarket([(50, 2), (10, 1), (20, 2), (30, 1)]) == 80:
        print('passed')
    else:
        print('failed')
```

```
    if supermarket([(100, 2), (10, 1), (15, 2), (20, 1), (1, 3)]) == 130:
        print('passed')
    else:
        print('failed')

    if supermarket([(5, 1), (7, 1), (8, 1)]) == 8:
        print('passed')
    else:
        print('failed')

    if supermarket([(1, 5), (10, 3), (1, 3)]) == 11:
        print('passed')
    else:
        print('failed')

if __name__ == '__main__':
    test_suite()
```

Output:



Final output of the whole code: