Started on	Tuesday, 22 October 2024, 3:17 PM
State	Finished
Completed on	Tuesday, 22 October 2024, 3:50 PM
Time taken	32 mins 49 secs
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Given a 2D matrix **tsp[][]**, where each row has the array of distances from that indexed city to all the other cities and **-1** denotes that there doesn't exist a path between those two indexed cities. The task is to print minimum cost in TSP cycle.

tsp[][] = {{-1, 30, 25, 10},
{15, -1, 20, 40},
{10, 20, -1, 25},
{30, 10, 20, -1}};

Answer: (penalty regime: 0 %)

Reset answer

```python
from typing import DefaultDict
INT_MAX = 2147483647
def findMinRoute(tsp):
    sum = 0
    counter = 0
    j = 0
    i = 0
    min = INT_MAX
    visitedRouteList = DefaultDict(int)
    visitedRouteList[0] = 1
    route = [0] * len(tsp)
    while i < len(tsp) and j < len(tsp[i]):
        #Write your code here
        #Start here
        if counter >= len(tsp[i]) - 1:
            break
        if j != i and (visitedRouteList[j] == 0):
            if tsp[i][j] < min:
                min = tsp[i][j]
                route[counter] = j + 1
        j += 1
        if j == len(tsp[i]):
```

	Expected	Got	
✓	`Minimum Cost is : 50`	`Minimum Cost is : 50`	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

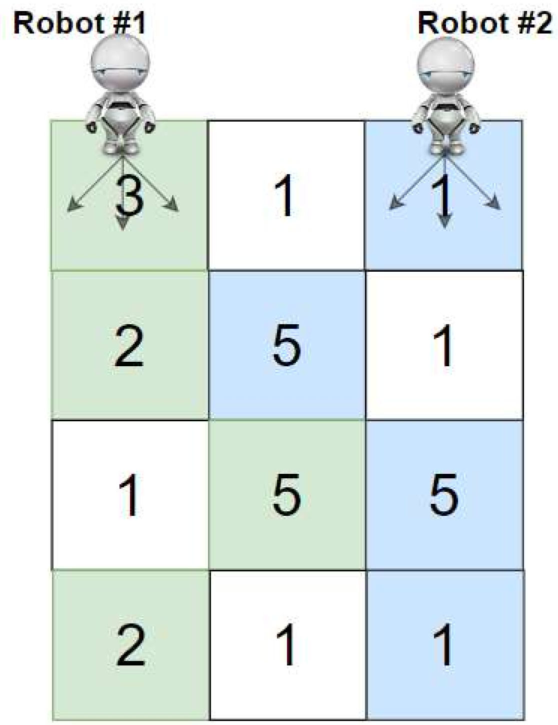
You are given a `rows x cols` matrix `grid` representing a field of cherries where `grid[i][j]` represents the number of cherries that you can collect from the `(i, j)` cell.

You have two robots that can collect cherries for you:

- **Robot #1** is located at the **top-left corner** `(0, 0)`, and
- **Robot #2** is located at the **top-right corner** `(0, cols - 1)`.

Return *the maximum number of cherries collection using both robots by following the rules below*:

- From a cell `(i, j)`, robots can move to cell `(i + 1, j - 1)`, `(i + 1, j)`, or `(i + 1, j + 1)`.
- When any robot passes through a cell, It picks up all cherries, and the cell becomes an empty cell.
- When both robots stay in the same cell, only one takes the cherries.
- Both robots cannot move outside of the grid at any moment.
- Both robots should reach the bottom row in `grid`.



3	1	1
2	5	1
1	5	5
2	1	1

For example:

Test	Result
`ob.cherryPickup(grid)`	24

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution(object):
    def cherryPickup(self, grid):
        dp=[[0 for j in range(len(grid))]for i in range(len(grid))]
        for i in range(len(grid)):
            for j in range(len(grid)-1):
                dp[i][j]=grid[i-1][j-1]
        res=len(grid)*6


        ROW_NUM = len(grid)
        COL_NUM = len(grid[0])
        return dp[0][COL_NUM - 1]*res

grid=[[3,1,1],
      [2,5,1],
      [1,5,5],
      [2,1,1]]
ob=Solution()
print(ob.cherryPickup(grid))

```

	Test	Expected	Got	
✓	ob.cherryPickup(grid)	24	24	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Create a python program using dynamic programming for 0/1 knapsack problem.

For example:

Test	Input	Result
knapSack(W, wt, val, n)	3 3 50 60 100 120 10 20 30	The maximum value that can be put in a knapsack of capacity W is: 220

Answer: (penalty regime: 0 %)

Reset answer

```python
def knapSack(W, wt, val, n):
    #start
    if n==0 or W==0:
        return 0
    if wt[n-1]>W:
        return knapSack(W,wt,val,n-1)
    else:
        return max(val[n-1]+knapSack(W-wt[n-1],wt,val,n-1),knapSack(W,wt,val,n-1))

x=int(input())
y=int(input())
W=int(input())
val=[]
wt=[]
for i in range(x):
    val.append(int(input()))
for y in range(y):
    wt.append(int(input()))

n = len(val)
print('The maximum value that can be put in a knapsack of capacity W is: ',knapSack(W, wt, val, n))

```

	Test	Input	Expected	Got	
✓	knapSack(W, wt, val, n)	3 3 50 60 100 120 10 20 30	The maximum value that can be put in a knapsack of capacity W is: 220	The maximum value that can be put in a knapsack of capacity W is: 220	✓

	Test	Input	Expected	Got	
✓	knapSack(W, wt, val, n)	3 3 40 50 90 110 10 20 30	The maximum value that can be put in a knapsack of capacity W is: 160	The maximum value that can be put in a knapsack of capacity W is: 160	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Incorrect

Mark 0.00 out of 20.00

Create a python program to find the Hamiltonian path using Depth First Search for traversing the graph .

For example:

Test	Result

`hamiltonian.findCycle()`	`['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'A']`
	`['A', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A']`

Answer: (penalty regime: 0 %)

Reset answer

```
 1 class Hamiltonian:
 2     def __init__(self, start):
 3         self.start = start
 4         self.cycle = []
 5         self.hasCycle = False
 6 
 7     def findCycle(self):
 8         self.cycle.append(self.start)
 9         self.solve(self.start)
10 
11     def solve(self, vertex):
12         ############  Add your code here ##############
13 
14 
15 
16     def displayCycle(self):
17         names = []
18         for v in self.cycle:
19             names.append(vertices[v])
20         print(names)
21 
22 
```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 16)
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Create a python program using brute force method of searching for the given substring in the main string.

For example:

Test	Input	Result
`match(str1,str2)`	AABAACAADAABAABA AABA	Found at index 0 Found at index 9 Found at index 12

Answer: (penalty regime: 0 %)

Reset answer

```python
import re #Import this package
def match(str1,str2):

    pattern = re.compile(str2)
    r = pattern.search(str1)
    while r:
        print("Found at index {}".format(r.start()))
        r = pattern.search(str1,r.start() + 1)

str1=input()
str2=input()
```

	Test	Input	Expected	Got	
✓	`match(str1,str2)`	AABAACAADAABAABA AABA	Found at index 0 Found at index 9 Found at index 12	Found at index 0 Found at index 9 Found at index 12	✓
✓	`match(str1,str2)`	saveetha savee	Found at index 0	Found at index 0	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.