Started on	Friday, 9 May 2025, 2:13 PM
State	Finished
Completed on	Friday, 9 May 2025, 2:43 PM
Time taken	29 mins 45 secs
Grade	**100.00** out of 100.00

Question **1**

Incorrect

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

```
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
✖	Minimum Cost is : 50

Your code must pass all tests to earn any marks. Try again.

Incorrect

Marks for this submission: 0.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

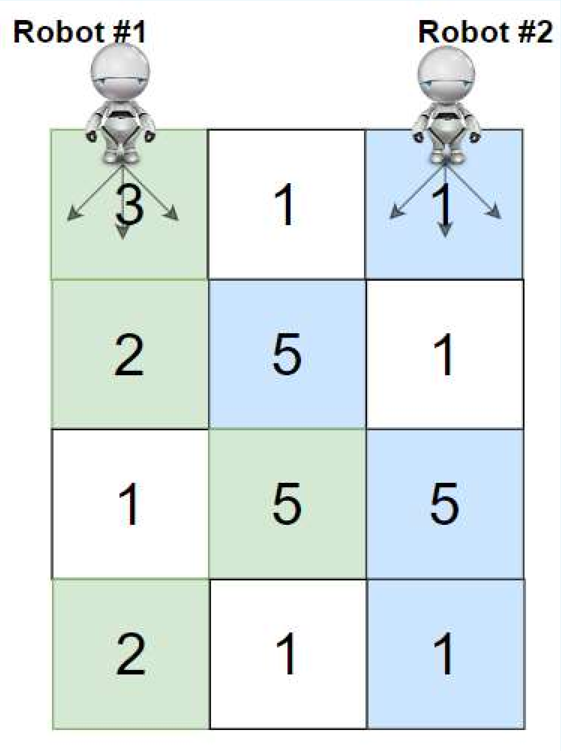
You are given a `rows x cols` matrix `grid` representing a field of cherries where `grid[i][j]` represents the number of cherries that you can collect from the `(i, j)` cell.

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):
        def dp(k):
            if k == ROW_NUM - 1:
                return [[grid[-1][i] if i == j else grid[-1][i] + grid[-1][j] for j in range(COL_NUM)
                        for i in range(COL_NUM)]
            row = grid[k]
            ans = [[0] * COL_NUM for i in range(COL_NUM)]
            next_dp = dp(k + 1)
            for i in range(COL_NUM):
                for j in range(i, COL_NUM):
                    for di in [-1, 0, 1]:
                        for dj in [-1, 0, 1]:
                            if 0 <= i + di < COL_NUM and 0 <= j + dj < COL_NUM:
                                if i == j:
                                    ans[i][j] = max(ans[i][j], next_dp[i + di][j + dj] + row[i])
                                else:
                                    ans[i][j] = max(ans[i][j], next_dp[i + di][j + dj] + row[i] + row
            return ans
        ROW_NUM = len(grid)
        COL_NUM = len(grid[0])
```

	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):
    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**

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
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)
    #End here
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.

Question **5**

Correct

Mark 20.00 out of 20.00

Write a Python program to sort unsorted numbers using Multi-key quicksort

For example:

Test	Input	Result
quick_sort_3partition(nums, 0, len(nums)-1)	5 4 3 5 1 2	Original list: [4, 3, 5, 1, 2] After applying Random Pivot Quick Sort the said list becomes: [1, 2, 3, 4, 5]
quick_sort_3partition(nums, 0, len(nums)-1)	6 21 10 3 65 4 8	Original list: [21, 10, 3, 65, 4, 8] After applying Random Pivot Quick Sort the said list becomes: [3, 4, 8, 10, 21, 65]

Answer: (penalty regime: 0 %)

```python
def quick_sort_3partition(arr,l,r):
    if r-l>1:
        p=partition(arr,l,r)
        quick_sort_3partition(arr,l,p)
        quick_sort_3partition(arr,p+1,r)
def partition(arr,l,r):
    pivot=arr[l]
    i=l+1
    j=r-1
    while True:
        while i<=j and arr[i]<=pivot:
            i=i+1
        while i<=j and arr[j]>=pivot:
            j=j-1
        if i<=j:
            arr[i],arr[j]=arr[j],arr[i]
        else:
            arr[l],arr[j]=arr[j],arr[l]
            return j
nums=[]
n=int(input())
for i in range(n):
```

	Test	Input	Expected	Got	
✓	quick_sort_3partition(nums, 0, len(nums)-1)	5 4 3 5 1 2	Original list: [4, 3, 5, 1, 2] After applying Random Pivot Quick Sort the said list becomes: [1, 2, 3, 4, 5]	Original list: [4, 3, 5, 1, 2] After applying Random Pivot Quick Sort the said list becomes: [1, 2, 3, 4, 5]	✓

	Test	Input	Expected	Got	
✓	quick_sort_3partition(nums, 0, len(nums)-1)	6 21 10 3 65 4 8	Original list: [21, 10, 3, 65, 4, 8] After applying Random Pivot Quick Sort the said list becomes: [3, 4, 8, 10, 21, 65]	Original list: [21, 10, 3, 65, 4, 8] After applying Random Pivot Quick Sort the said list becomes: [3, 4, 8, 10, 21, 65]	✓
✓	quick_sort_3partition(nums, 0, len(nums)-1)	4 21 3 10 4	Original list: [21, 3, 10, 4] After applying Random Pivot Quick Sort the said list becomes: [3, 4, 10, 21]	Original list: [21, 3, 10, 4] After applying Random Pivot Quick Sort the said list becomes: [3, 4, 10, 21]	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.