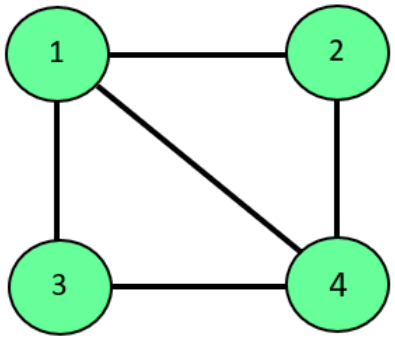
Started on	Monday, 12 May 2025, 9:23 AM
State	Finished
Completed on	Monday, 12 May 2025, 9:57 AM
Time taken	34 mins 52 secs
Grade	**80.00** out of 100.00

Question **1**

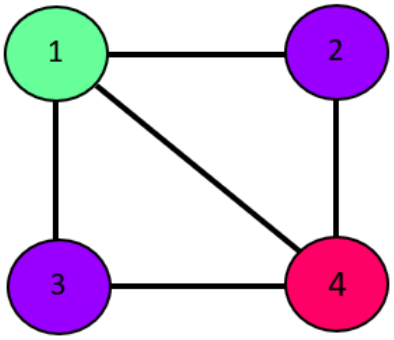
Incorrect

Mark 0.00 out of 20.00

The m-coloring problem states, **"We are given an undirected graph and m number of different colors. We have to check if we can assign colors to the vertices of the graphs in such a way that no two adjacent vertices have the same color."**



0	1	1	1
1	0	0	1
1	0	0	1
1	1	1	0



Node 1 -> color 1
Node 2 -> color 2
Node 3 -> color 2
Node 4-> color 3

For example:

Result
Solution Exists: Following are the assigned colors Vertex 1 is given color: 1 Vertex 2 is given color: 2 Vertex 3 is given color: 3 Vertex 4 is given color: 2

Answer: (penalty regime: 0 %)

Reset answer

```
def isSafe(graph, color):
    for i in range(4):
        for j in range(i + 1, 4):
            if (graph[i][j] and color[j] == color[i]):
                return False
    return True

def graphColoring(graph, m, i, color):

    ############ Add your code here ####################
def display(color):
    print("Solution Exists:" " Following are the assigned colors ")
    for i in range(4):
        print("Vertex", i+1 ," is given color: ",color[i])
if __name__ == '__main__':
    graph = [
        [ 0, 1, 1, 1 ],
        [ 1, 0, 1, 0 ],
        [ 1, 1, 0, 1 ],
        [ 1, 0, 1, 0 ],
    ]
    m = 3 # Number of colors
```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 11)
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Create a python Program to find the maximum contiguous sub array using Dynamic Programming.

For example:

Test	Input	Result
maxSubArraySum(a,len(a))	8 -2 -3 4 -1 -2 1 5 -3	Maximum contiguous sum is 7

Answer: (penalty regime: 0 %)

```
def maxSubArraySum(a,size):
    cur_sum = a[0]
    max_sum = a[0]
    for i in range(1,size):
        cur_sum += a[i]
        if cur_sum < 0:
            cur_sum = 0
        if cur_sum > max_sum:
            max_sum = cur_sum
    print("Maximum contiguous sum is",max_sum)
size = int(input())
a = []
for i in range(size):
    a.append(int(input()))
```

	Test	Input	Expected	Got	
✔	maxSubArraySum(a,len(a))	8 -2 -3 4 -1 -2 1 5 -3	Maximum contiguous sum is 7	Maximum contiguous sum is 7	✔
✔	maxSubArraySum(a,len(a))	5 1 2 3 -4 -6	Maximum contiguous sum is 6	Maximum contiguous sum is 6	✔

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 to find Minimum number of jumps to reach end of the array using naive method(recursion)

For example:

Test	Input	Result
minJumps(arr, 0, n-1)	10 1 3 6 3 2 3 6 8 9 5	Minimum number of jumps to reach end is 4

Answer: (penalty regime: 0 %)

Reset answer

```
def minJumps(arr, l, h):
    if l==h:
        return 0
    if arr[l] == 0:
        return float('inf')
    min_jum = float('inf')
    for i in range(l+1,h+1):
        if l+arr[l] >= i:
            jumps = minJumps(arr,i,h)
            if jumps != float('inf') and jumps+1 < min_jum:
                min_jum = jumps+1
    return min_jum


arr = []
n = int(input())
for i in range(n):
    arr.append(int(input()))
print('Minimum number of jumps to reach','end is', minJumps(arr, 0, n-1))
```

	Test	Input	Expected	Got	
✔	minJumps(arr, 0, n-1)	10 1 3 6 3 2 3 6 8 9 5	Minimum number of jumps to reach end is 4	Minimum number of jumps to reach end is 4	✔
✔	minJumps(arr, 0, n-1)	7 3 2 5 9 4 1 6	Minimum number of jumps to reach end is 2	Minimum number of jumps to reach end is 2	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Write a python program to Implement Minimum cost path using Dynamic Programming.

For example:

Input	Result
3 3	8

Answer: (penalty regime: 0 %)

```
r = int(input())
c = int(input())

def min_cost_path(cost,m,n):
    dp = [[0 for i in range(c)]for j in range(r)]
    dp[0][0] = cost[0][0]
    for i in range(1,r):
        dp[0][i] = dp[0][i-1] + cost[0][i]
    for j in range(1,c):
        dp[j][0] = dp[j-1][0] + cost[j][0]
    for i in range(1,r):
        for j in range(1,c):
            dp[i][j] = min(dp[i-1][j],dp[i][j-1],dp[i-1][j-1]) + cost[i][j]
    return dp[m][n]

cost = [ [1,2,3],[4,8,2],[1,5,3]]
print(min_cost_path(cost,r-1,c-1))
```

	Input	Expected	Got	
✔	3 3	8	8	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Create a Dynamic Programming python Implementation of Coin Change Problem.

For example:

Test	Input	Result
count(arr, m, n)	3 4 1 2 3	4

Answer: (penalty regime: 0 %)

Reset answer

```
def count(S, m, n):
    table = [[0 for x in range(m)] for x in range(n+1)]
    for i in range(m):
        table[0][i] = 1
    for i in range(1, n+1):
        for j in range(m):

            # Count of solutions including S[j]
            if S[j] <= i:
                incl = table[i-S[j]][j]
            else:
                incl =0

            # Count of solutions excluding S[j]
            if j>0:
               exc = table [i][j-1]
            else:
                exc = 0

            # total count
            table[i][j] = incl + exc

```

	Test	Input	Expected	Got	
✔	count(arr, m, n)	3 4 1 2 3	4	4	✔
✔	count(arr, m, n)	3 16 1 2 5	20	20	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.