Started on	Monday, 23 September 2024, 9:10 AM
State	Finished
Completed on	Monday, 23 September 2024, 9:36 AM
Time taken	25 mins 12 secs
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a Program to Create a recursive function to reverse a string.

For example:

Input	Result
desserts	stressed
knits	stink

Answer: (penalty regime: 0 %)

```
def reverse(n):
    if n>=0:
        print(string[n],end="")
        reverse(n-1)
string=input()
reverse(len(string)-1)
```

	Input	Expected	Got	
✔	desserts	stressed	stressed	✔
✔	knits	stink	stink	✔
✔	Regal	lageR	lageR	✔
✔	pupils	slipup	slipup	✔
✔	smart	trams	trams	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Write a Python Program for printing Minimum Cost Simple Path between two given nodes in a directed and weighted graph

For example:

Test	Result
`minimumCostSimplePath(s, t, visited, graph)`	`-3`

Answer: (penalty regime: 0 %)

Reset answer

```
import sys
V = 5
INF = sys.maxsize
def minimumCostSimplePath(u, destination,
                          visited, graph):
########  Add your code here ##############
    #Start here
    if (u == destination):
        return 0
    visited[u] = 1
    ans = INF
    for i in range(V):
        if (graph[u][i] != INF and not visited[i]):
            curr = minimumCostSimplePath(i, destination,
                                         visited, graph)
            if (curr < INF):
                ans = min(ans, graph[u][i] + curr)
    visited[u] = 0
    return ans
    #End here

if __name__=="__main__":
```

	Test	Expected	Got	
✔	`minimumCostSimplePath(s, t, visited, graph)`	`-3`	`-3`	✔

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 function to compute the fewest number of coins that we need to make up the amount given.

For example:

Test	Input	Result
ob1.coinChange(s,amt)	3 11 1 2 5	3

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution(object):
    def coinChange(self, coins, amount):
        ####################      Add your Code Here ###########
        #End here
        if amount == 0 :
            return 0
        if min(coins) > amount:
            return -1
        dp = [-1 for i in range(0, amount + 1)]
        for i in coins:
            if i > len(dp) - 1:
                continue
            dp[i] = 1
            for j in range(i + 1, amount + 1):
                if dp[j - i] == -1:
                    continue
                elif dp[j] == -1:
                    dp[j] = dp[j - i] + 1
                else:
                    dp[j] = min(dp[j], dp[j - i] + 1)
        return dp[amount]
    #End here
```

	Test	Input	Expected	Got	
✔	ob1.coinChange(s,amt)	3 11 1 2 5	3	3	✔
✔	ob1.coinChange(s,amt)	3 12 1 2 5	3	3	✔
✔	ob1.coinChange(s,amt)	3 22 1 2 5	5	5	✔

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 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):
    max_so_far = a[0]
    max_ending_here = 0
    for i in range(0, size):
        max_ending_here = max_ending_here + a[i]
        if max_ending_here < 0:
            max_ending_here = 0
        elif (max_so_far < max_ending_here):
            max_so_far = max_ending_here

    return max_so_far
n=int(input())
a =[]
for i in range(n):
    a.append(int(input()))
print("Maximum contiguous sum is", maxSubArraySum(a,n))
```

	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 **5**

Correct

Mark 20.00 out of 20.00

Print All Paths With Minimum Jumps

```
1. You are given a number N representing number of elements.
2. You are given N space separated numbers (ELE : elements).
3. Your task is to find & print
    3.1) "MINIMUM JUMPS" need from 0th step to (n-1)th step.
    3.2) all configurations of "MINIMUM JUMPS".
NOTE: Checkout sample question/solution video inorder to have more insight.
```

For example:

Test	Input	Result
minJumps(arr)	10 3 3 0 2 1 2 4 2 0 0	0 -> 3 -> 5 -> 6 -> 9 0 -> 3 -> 5 -> 7 -> 9

Answer: (penalty regime: 0 %)

Reset answer

```
from queue import Queue
import sys
class Pair(object):
    idx = 0
    psf = ""
    jmps = 0
    def __init__(self, idx, psf, jmps):

        self.idx = idx
        self.psf = psf
        self.jmps = jmps
def minJumps(arr):
    ##################  Add your Code here.
    #Start here
    MAX_VALUE = sys.maxsize
    dp = [MAX_VALUE for i in range(len(arr))]
    n = len(dp)
    dp[n - 1] = 0
    for i in range(n - 2, -1, -1):
        steps = arr[i]
        minimum = MAX_VALUE
        for j in range(1, steps + 1, 1):
```

	Test	Input	Expected	Got	
✔	minJumps(arr)	10 3 3 0 2 1 2 4 2 0 0	0 -> 3 -> 5 -> 6 -> 9 0 -> 3 -> 5 -> 7 -> 9	0 -> 3 -> 5 -> 6 -> 9 0 -> 3 -> 5 -> 7 -> 9	✔

	Test	Input	Expected	Got	
✔	minJumps(arr)	7 5 5 0 3 2 3 6	0 -> 1 -> 6 0 -> 3 -> 6 0 -> 4 -> 6 0 -> 5 -> 6	0 -> 1 -> 6 0 -> 3 -> 6 0 -> 4 -> 6 0 -> 5 -> 6	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.