PRACTICAL NO. 6

Name	Shreyash Yadav
Section and Batch	A4 - B3
Roll no.	41

Aim: Construction of OBST

Problem Statement: Smart Library Search Optimization
Task 1:
Scenario:
A university digital library system stores frequently accessed books using a binary search mechanism. The library admin wants to minimize the average search time for book lookups by arranging the book IDs optimally in a binary search tree.

Each book ID has a probability of being searched successfully and an associated probability for unsuccessful searches (when a book ID does not exist between two keys).

Your task is to determine the minimum expected cost of searching using an Optimal Binary Search Tree (OBST).

Code:

```
def obst(p, q):
    n = len(p)
    p1 = [0.0] + p
    e = [[0.0] * (n + 2) for _ in range(n + 2)]
    w = [[0.0] * (n + 2) for _ in range(n + 2)]
    root = [[0] * (n + 2) for _ in range(n + 2)]

    for i in range(1, n + 2):
        e[i][i - 1] = q[i - 1]
        w[i][i - 1] = q[i - 1]

    for l in range(1, n + 1):
        for i in range(1, n - l + 2):
            j = i + l - 1
            e[i][j] = float('inf')
            w[i][j] = w[i][j - 1] + p1[j] + q[j]
            for r in range(i, j + 1):
                cost = e[i][r - 1] + e[r + 1][j] + w[i][j]
                if cost < e[i][j]:
```

```
                    e[i][j] = cost
                    root[i][j] = r

    def preorder(i, j):
        if i > j:
            return []
        r = root[i][j]
        return [r - 1] + preorder(i, r - 1) + preorder(r + 1, j)

    min_cost = round(e[1][n], 4)
    preorder_traversal = preorder(1, n)
    return min_cost, preorder_traversal


p = [0.1, 0.2, 0.4, 0.3]
q = [0.05, 0.1, 0.05, 0.05, 0.1]

cost, preorder_trav = obst(p, q)
print(f"Minimum Expected Cost: {cost:.4f}")
print(f"Preorder Traversal of OBST: {preorder_trav}")
```

Output:

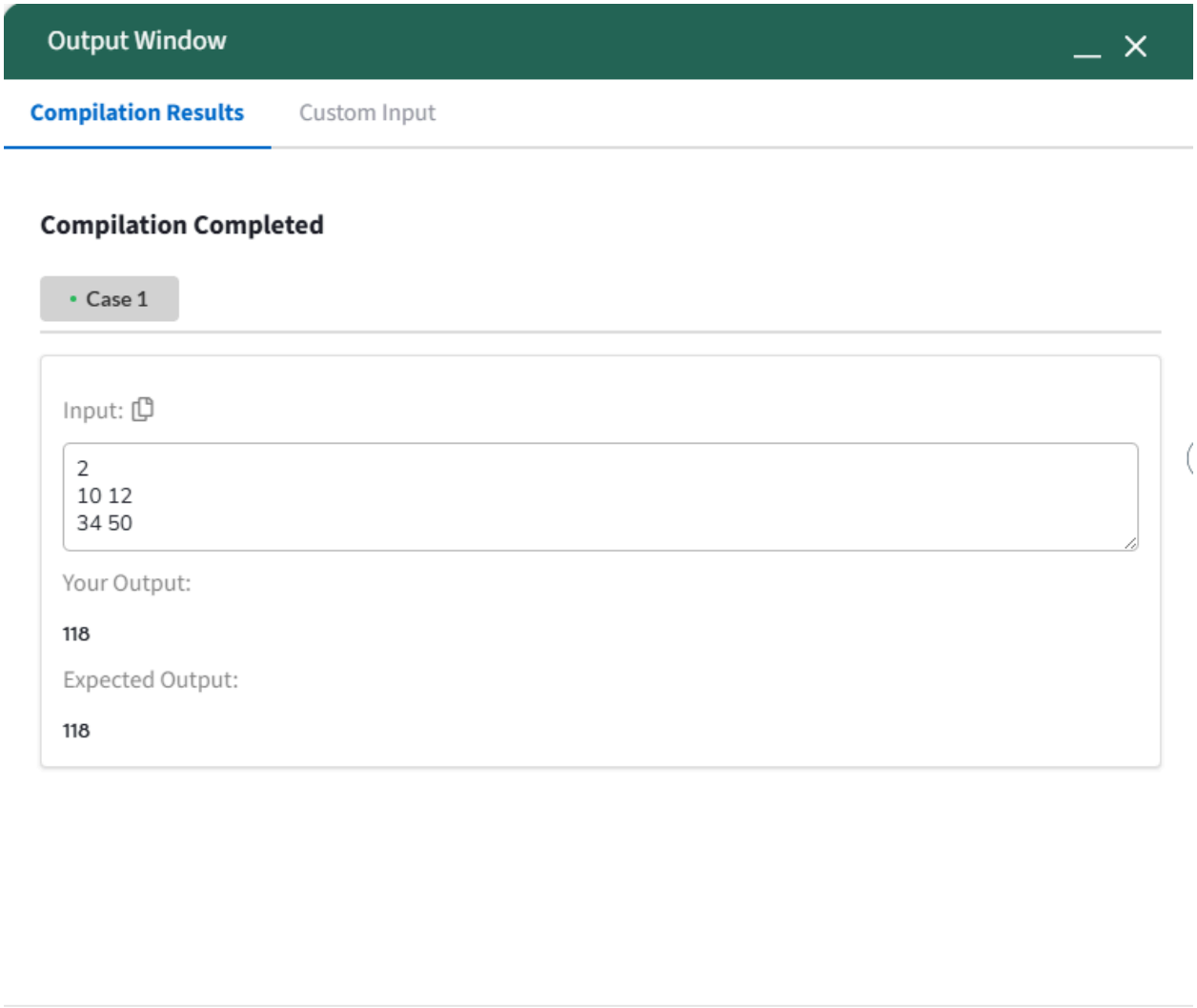
```
Minimum Expected Cost: 2.9000
Preorder Traversal of OBST: [2, 1, 0, 3]
```

Task 2:
https://www.geeksforgeeks.org/problems/optimal-binary-search-tree2214/1

Code:

```python
# User function Template for python3

class Solution:
    def optimalSearchTree(self, keys, freq, n):
        cost = [[0] * n for _ in range(n)]

        for i in range(n):
            cost[i][i] = freq[i]


        for l in range(2, n + 1):
            for i in range(n - l + 1):
                j = i + l - 1
                cost[i][j] = float('inf')
                total = sum(freq[i:j + 1])

                for r in range(i, j + 1):
                    left = cost[i][r - 1] if r > i else 0
                    right = cost[r + 1][j] if r < j else 0
                    val = left + right + total
                    if val < cost[i][j]:
                        cost[i][j] = val

        return cost[0][n - 1]

```

Output:


Output Window
Compilation Results
Custom Input
Compilation Completed
Case 1
Input:
2
10 12
34 50
Your Output:
118
Expected Output:
118

Output Window

Compilation Results | Custom Input | Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed

104 / 104

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

8 / 8

Your Total Score: 8

Time Taken

1.12

Solve Next

Fixing Two nodes of a BST | Strictly Increasing Array | Word Wrap

Stay Ahead With:

Build 21 Projects in 21 Days

Build real-world ML, Deep Learning & Gen AI projects

Register Now

Kick start your career with GfG 160!