PRACTICAL NO. 6

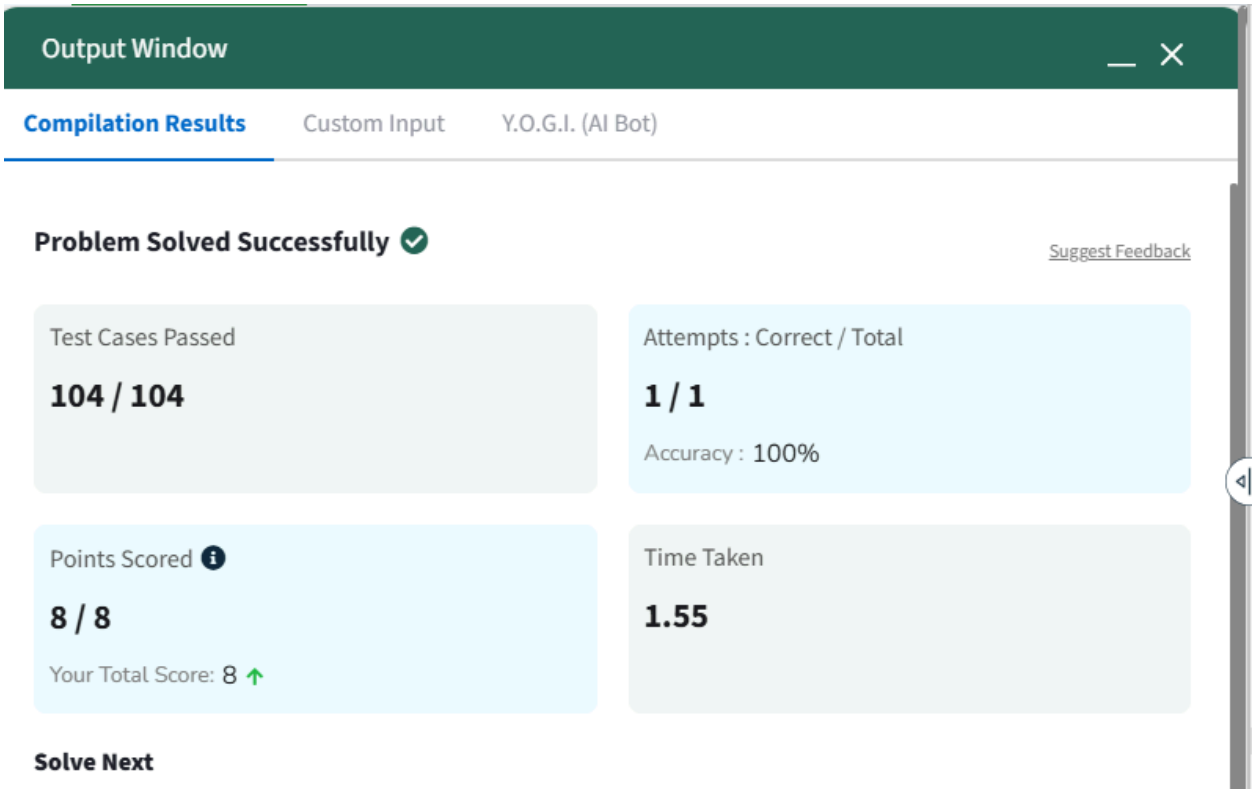
Problem Statement: Smart Library Search Optimization

Task 1:

```
def optimal_bst(p, q, n):
    e = [[0] * (n+2) for _ in range(n+2)]
    w = [[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 length in range(1, n+1):
        for i in range(1, n-length+2):
            j = i + length - 1
            e[i][j] = float('inf')
            w[i][j] = w[i][j-1] + p[j-1] + 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

    return e[1][n]


n = 4
p = [0.1, 0.2, 0.4, 0.3]
q = [0.05, 0.1, 0.05, 0.05, 0.1]

result = optimal_bst(p, q, n)
print(f"{result:.4f}")
```

```
2.9000
```

Task 2:


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
1.55
Solve Next

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

        for i in range(n):
            sumFreq[i][i] = freq[i]
            for j in range(i+1, n):
                sumFreq[i][j] = sumFreq[i][j-1] + freq[j]

        for length in range(1, n+1):
            for i in range(n - length + 1):
                j = i + length - 1
                if i == j:
                    dp[i][j] = freq[i]
                else:
                    dp[i][j] = float('inf')
                    for r in range(i, j+1):
                        cost_left = dp[i][r-1] if r > i else 0
                        cost_right = dp[r+1][j] if r < j else 0
                        cost = cost_left + cost_right + sumFreq[i][j]
                        if cost < dp[i][j]:
                            dp[i][j] = cost

        return dp[0][n-1]
```