Practical 6

Name	Dipansh Tiwari
Section	A4
Roll no	B3-35

Task 1

Code-

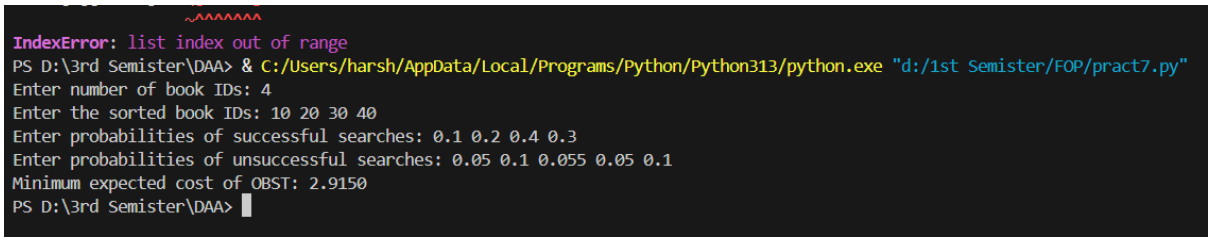
```
def optimal_bst(keys, p, q, n):
    e = [[0 for _ in range(n + 2)] for _ in range(n + 2)]
    w = [[0 for _ in range(n + 2)] for _ in range(n + 2)]
    for i in range(1, n + 2):
        e[i][i - 1] = q[i - 1]
        w[i][i - 1] = q[i - 1]
    for l in range(1, n + 1):
        for i in range(1, n - l + 2):
            j = i + l - 1
            e[i][j] = float('inf')
            w[i][j] = w[i][j - 1] + 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 = int(input("Enter number of book IDs: "))
keys = list(map(int, input("Enter the sorted book IDs: ").split()))
p = list(map(float, input("Enter probabilities of successful searches: ").split()))
q = list(map(float, input("Enter probabilities of unsuccessful searches: ").split()))
```

min_cost = optimal_bst(keys, p, q, n)

print(f"Minimum expected cost of OBST: {min_cost:.4f}")

Output-



Task 2-

Code-

```
class Solution
{
   static int optimalSearchTree(int keys[], int freq[], int n)
   {
      int cost[][] = new int[n][n];
      int sum[][] = new int[n][n];

      for (int i = 0; i < n; i++)
      {
         cost[i][i] = freq[i];
         sum[i][i] = freq[i];
      }

      for (int len = 2; len <= n; len++)
      {
         for (int i = 0; i <= n - len; i++)
```

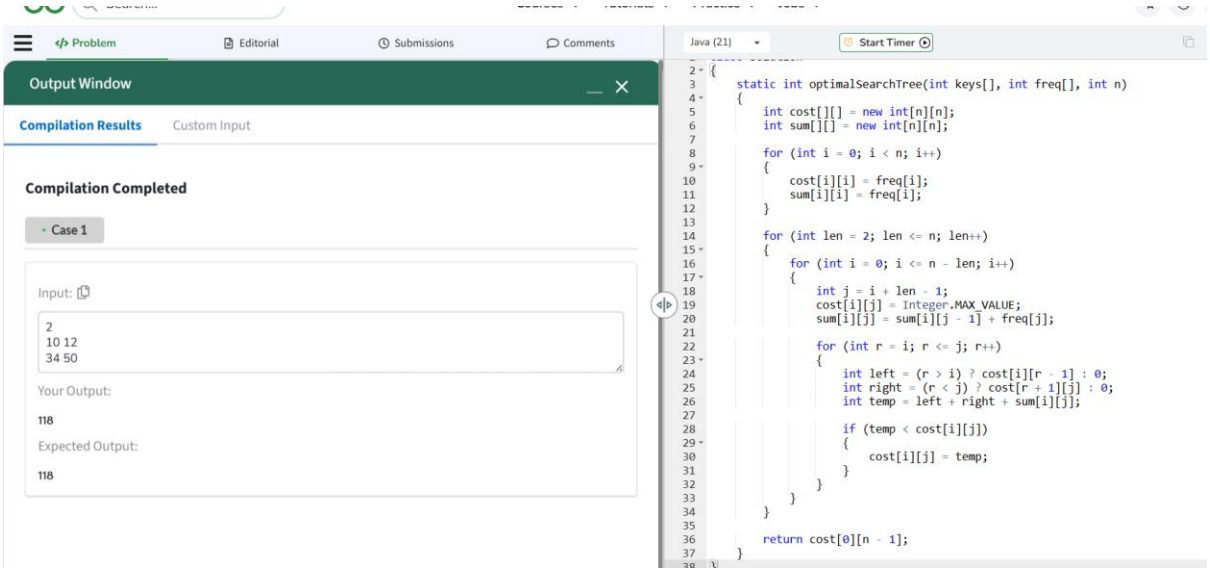
```
        {
            int j = i + len - 1;
            cost[i][j] = Integer.MAX_VALUE;
            sum[i][j] = sum[i][j - 1] + freq[j];

            for (int r = i; r <= j; r++)
            {
                int left = (r > i) ? cost[i][r - 1] : 0;
                int right = (r < j) ? cost[r + 1][j] : 0;
                int temp = left + right + sum[i][j];

                if (temp < cost[i][j])
                {
                    cost[i][j] = temp;
                }
            }
        }
    }

    return cost[0][n - 1];
  }
}
```

Output-


Problem
Editorial
Submissions
Comments
Output Window
Compilation Results
Custom Input
Compilation Completed
Case 1
Input:
2
10 12
34 50
Your Output:
118
Expected Output:
118
Java (21)
Start Timer
{
static int optimalSearchTree(int keys[], int freq[], int n)
{
int cost[][] = new int[n][n];
int sum[][] = new int[n][n];
for (int i = 0; i < n; i++)
{
cost[i][i] = freq[i];
sum[i][i] = freq[i];
}
for (int len = 2; len <= n; len++)
{
for (int i = 0; i <= n - len; i++)
{
int j = i + len - 1;
cost[i][j] = Integer.MAX_VALUE;
sum[i][j] = sum[i][j - 1] + freq[j];
for (int r = i; r <= j; r++)
{
int left = (r > i) ? cost[i][r - 1] : 0;
int right = (r < j) ? cost[r + 1][j] : 0;
int temp = left + right + sum[i][j];
if (temp < cost[i][j])
{
cost[i][j] = temp;
}
}
}
}
return cost[0][n - 1];
}