Practical 6

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).

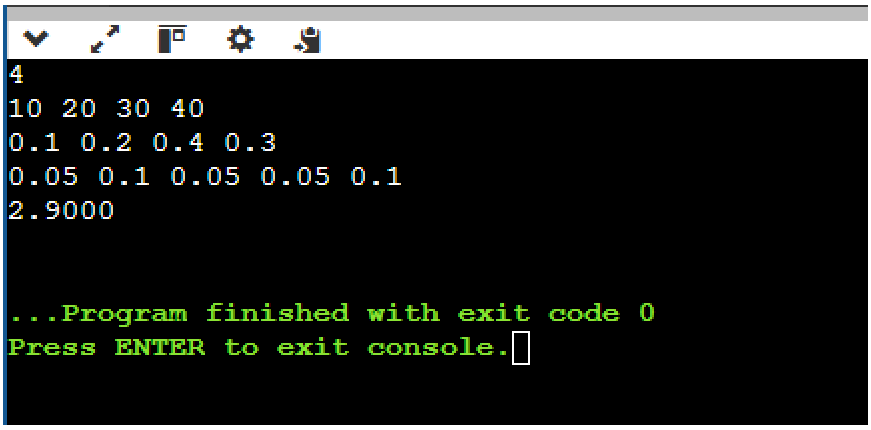
Your task is to determine the minimum expected cost of searching using an Optimal Binary Search Tree (OBST).

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <float.h>

#define MAX 100

int main() {
    int n;
    scanf("%d", &n);

    int keys[MAX];
    double p[MAX], q[MAX + 1];
    for (int i = 0; i < n; i++)
        scanf("%d", &keys[i]);
    for (int i = 0; i < n; i++)
```

```
        scanf("%lf", &p[i]);
    for (int i = 0; i <= n; i++)
        scanf("%lf", &q[i]);

    double e[MAX + 1][MAX + 1] = {0};
    double w[MAX + 1][MAX + 1] = {0};
    int root[MAX + 1][MAX + 1] = {0};

    for (int i = 0; i <= n; i++) {
        e[i][i] = q[i];
        w[i][i] = q[i];
    }

    for (int l = 1; l <= n; l++) {
        for (int i = 0; i <= n - l; i++) {
            int j = i + l;
            e[i][j] = DBL_MAX;
            w[i][j] = w[i][j - 1] + p[j - 1] + q[j];
            for (int r = i + 1; r <= j; r++) {
                double t = e[i][r - 1] + e[r][j] + w[i][j];
                if (t < e[i][j]) {
                    e[i][j] = t;
                    root[i][j] = r;
                }
            }
        }
    }

    printf("%.4lf\n", e[0][n]);
    return 0;
}
```

Output :



```
4
10 20 30 40
0.1 0.2 0.4 0.3
0.05 0.1 0.05 0.05 0.1
2.9000


...Program finished with exit code 0
Press ENTER to exit console.
```

Task 2:

Code :

```
#include <stdio.h>

#include <limits.h>


int main() {

  int n, i, j, k, l;

  int keys[100], freq[100];

  int cost[100][100];


  scanf("%d", &n);

  for (i = 0; i < n; i++) scanf("%d", &keys[i]);

  for (i = 0; i < n; i++) scanf("%d", &freq[i]);


  for (i = 0; i < n; i++)

    cost[i][i] = freq[i];


  for (l = 2; l <= n; l++) {

    for (i = 0; i <= n - l; i++) {

      j = i + l - 1;
```

```
            cost[i][j] = INT_MAX;

            int sum = 0;
            for (k = i; k <= j; k++)
                sum += freq[k];

            for (k = i; k <= j; k++) {
                int left = 0, right = 0;
                if (k > i) left = cost[i][k - 1];
                if (k < j) right = cost[k + 1][j];

                int c = left + right + sum;

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

    printf("%d\n", cost[0][n - 1]);
    return 0;
}
```