Name: Areeb Ansari Roll No:A3-B1-01

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 <float.h>


void OptimalBST(double p[], double q[], int n,

        double E[n + 1][n + 1],

        double W[n + 1][n + 1],

        int R[n + 1][n + 1]) {

  int i, j, k, d;
```

```
double cost;

// Step 1: Initialization
for (i = 0; i <= n; i++) {
  for (j = 0; j <= n; j++) {   // Initialize to zero
    E[i][j] = 0.0;
    W[i][j] = 0.0;
    R[i][j] = 0;
  }
}

for (i = 0; i <= n; i++) {
  E[i][i] = q[i];
  W[i][i] = q[i];
  R[i][i] = 0; // No root for empty tree
}

// Step 2: Compute for all intervals of increasing length
for (d = 1; d <= n; d++) {
  for (i = 0; i <= n - d; i++) {
    j = i + d;
    E[i][j] = DBL_MAX;
    W[i][j] = W[i][j - 1] + p[j] + q[j];

    // Step 3: Try each key as root
    for (k = i + 1; k <= j; k++) {
      cost = E[i][k - 1] + E[k][j] + W[i][j];
      if (cost < E[i][j]) {
```

```
                E[i][j] = cost;
                R[i][j] = k;
            }
        }
    }
  }
}

int main() {
  int n, i, j;

  printf("Enter the number of book IDs: ");
  scanf("%d", &n);

  int keys[n + 1];
  double p[n + 1], q[n + 1];
  double E[n + 1][n + 1], W[n + 1][n + 1];
  int R[n + 1][n + 1];

  printf("Enter %d sorted book IDs:\n", n);
  for (i = 1; i <= n; i++) {
    printf("Key %d: ", i);
    scanf("%d", &keys[i]);
  }

  printf("\nEnter %d probabilities of successful searches (p[i]):\n", n);
  for (i = 1; i <= n; i++) {
    printf("p[%d]: ", i);
```

```
        scanf("%lf", &p[i]);
    }

    printf("\nEnter %d probabilities of unsuccessful searches (q[i]):\n", n + 1);
    for (i = 0; i <= n; i++) {
        printf("q[%d]: ", i);
        scanf("%lf", &q[i]);
    }

    OptimalBST(p, q, n, E, W, R);

    printf("\nMinimum Expected Cost: %.4lf\n", E[0][n]);

    printf("\nRoot Matrix (R):\n");
    for (i = 0; i <= n; i++) {
        for (j = 0; j <= n; j++)
            printf("%3d ", R[i][j]);
        printf("\n");
    }

    return 0;
}
```

Output:

```
Enter the number of book IDs: 4
Enter 4 sorted book IDs:
Key 1: 10
Key 2: 20
Key 3: 30
Key 4: 40

Enter 4 probabilities of successful searches (p[i]):
p[1]: 0.1
p[2]: 0.2
p[3]: 0.4
p[4]: 0.3

Enter 5 probabilities of unsuccessful searches (q[i]):
q[0]: 0.05
q[1]: 0.1
q[2]: 0.05
q[3]: 0.05
q[4]: 0.1

Minimum Expected Cost: 2.9000

Root Matrix (R):
  0   1   2   2   3
  0   0   2   3   3
  0   0   0   3   3
  0   0   0   0   4
  0   0   0   0   0
```

Task2: