**Practical 6**

**Name –** Ishwari Deshmukh

**Class –** A4 (B3)

**Roll no. –** 45

**Aim**: Construction of OBST

**Task 1 :**

**Problem Statement:** Smart Library Search Optimization

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>

int main() {

int n;

scanf("%d", &n);

int keys[n];

double p[n + 1], q[n + 1];

for (int i = 1; i <= n; i++)

scanf("%d", &keys[i - 1]);

for (int i = 1; i <= n; i++)

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

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

scanf("%lf", &q[i]);

double e[n + 2][n + 1];

double w[n + 2][n + 1];

int root[n + 1][n + 1];

for (int i = 1; i <= n + 1; i++) {

e[i][i - 1] = q[i - 1];

w[i][i - 1] = q[i - 1];

}

for (int l = 1; l <= n; l++) {

for (int i = 1; i <= n - l + 1; i++) {

int j = i + l - 1;

e[i][j] = DBL\_MAX;

w[i][j] = w[i][j - 1] + p[j] + q[j];

for (int r = i; r <= j; r++) {

double t = e[i][r - 1] + e[r + 1][j] + w[i][j];

if (t < e[i][j]) {

e[i][j] = t;

root[i][j] = r;

}

}

}

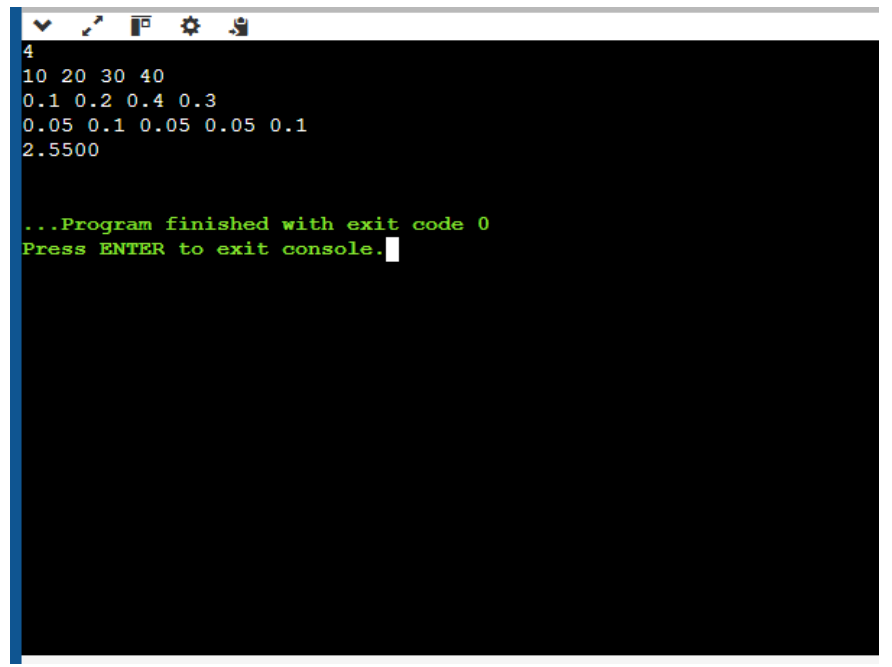
}

printf("Minimum Expected Cost: %.4f\n", e[1][n]);

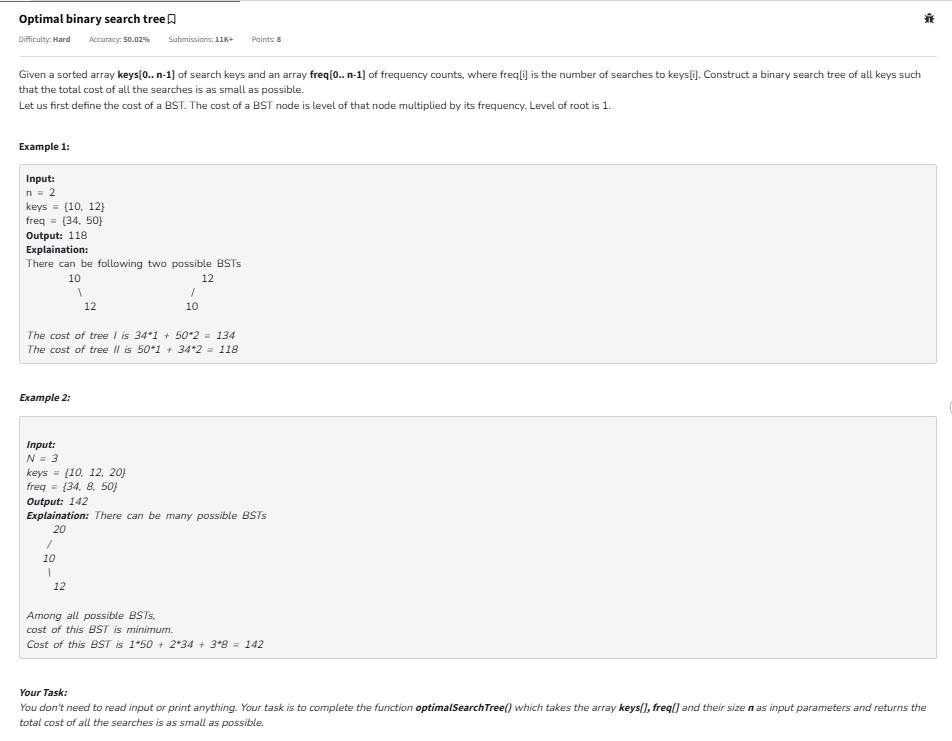
return 0;

}

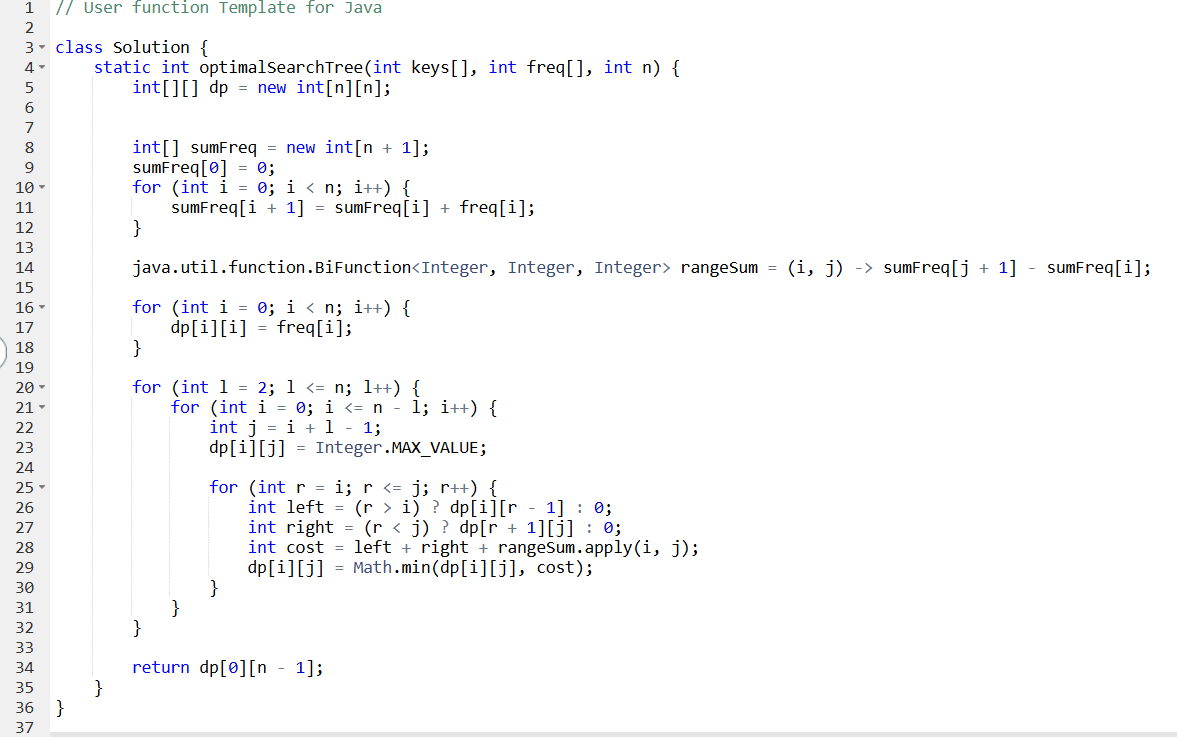
**Output-**

****

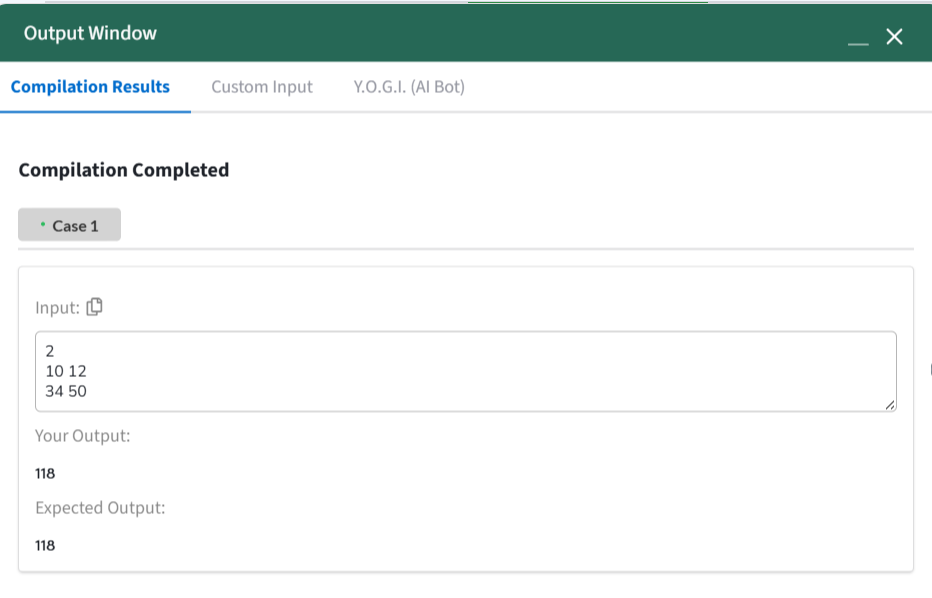
**Task 2:**

****

**Code –**

****

**Output –**

****