COMPUTER SOCIETY OF INDIA (CSI)
COEP TECH STUDENT CHAPTER

PRESENTS

CODE QUEST 5.0

EDITIORIAL

AMANDA THE FISHER GIRL

PROBLEM STATEMENT

In the scenic town of Aqualithia, there lives a fisher-girl Amanda who goes fishing every day. The rivers of Aqualithia are a home to various types of fishes and the fish types are denoted by unique random integers. Amanda needs to catch K fishes. There are consecutive fishing spots 0,1,2,3... and so on along the river marked at two meter increments starting from the 0th position. Amanda's house is situated at the 0th position. There are N fishes in the river. The i-th fish is at the Di fishing spot. Multiple fishes can be at the same fishing spot.

The type of the i-th fish is represented by Ti. The fish type is an important factor while fishing in Aqualithia as each fish type requires a unique fishing rod. Amanda has all the necessary fishing rods at her home but she can carry only one fishing rod at a time as these special fishing rods are very heavy.

PROBLEM STATEMENT

Amanda can catch a fish only if she is at the same fishing spot as the fish and uses the required type of fishing rod to catch the specific fish.

It takes Amanda one minute to go from the current fishing spot to the next or previous fishing spot and it takes no time to switch fishing rods when she is at home. Assume that when Amanda is at the appropriate fishing spot and carries the required fishing rod, it takes negligible time to catch the fish.

What is the least amount of time Amanda spends to catch K fish, and the least amount of distance she travels while doing this?

INPUT FORMAT

The first line of the input gives the number of test cases, T. T test cases follow. Each testcase begins with a line containing the two integers N and K, the number of fishes in the river and the number of fishes Amanda needs to catch, respectively.

The second line contains N integers, the i-th of which is Di, the fishing spot at which the i-th fish is present. The third line contains N integers, the i-th of which is Ti, the type of the i-th fish.

OUTPUT FORMAT & CONSTRAINTS

For each test case, output the sum of least amount of time and the least distance.

SAMPLE TESTCASES

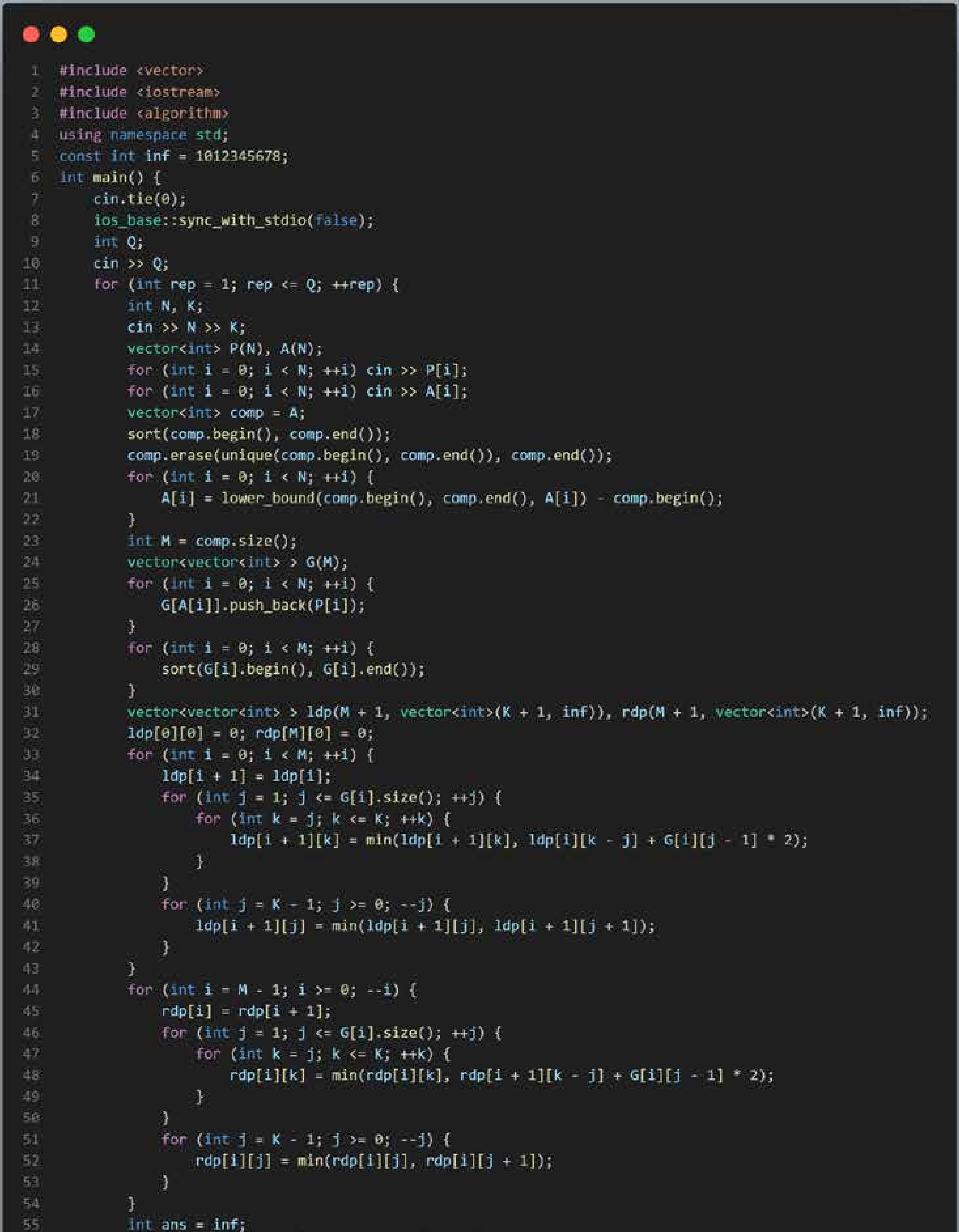
SAMPLE INPUT:

```
3
4 3
1 2 4 9
3 3 2 3
4 3
1 2 3 4
1 8 1 8
6 6
4 3 3 1 3 10000
1 2 8 9 5 7
```

SAMPLE OUTPUT:

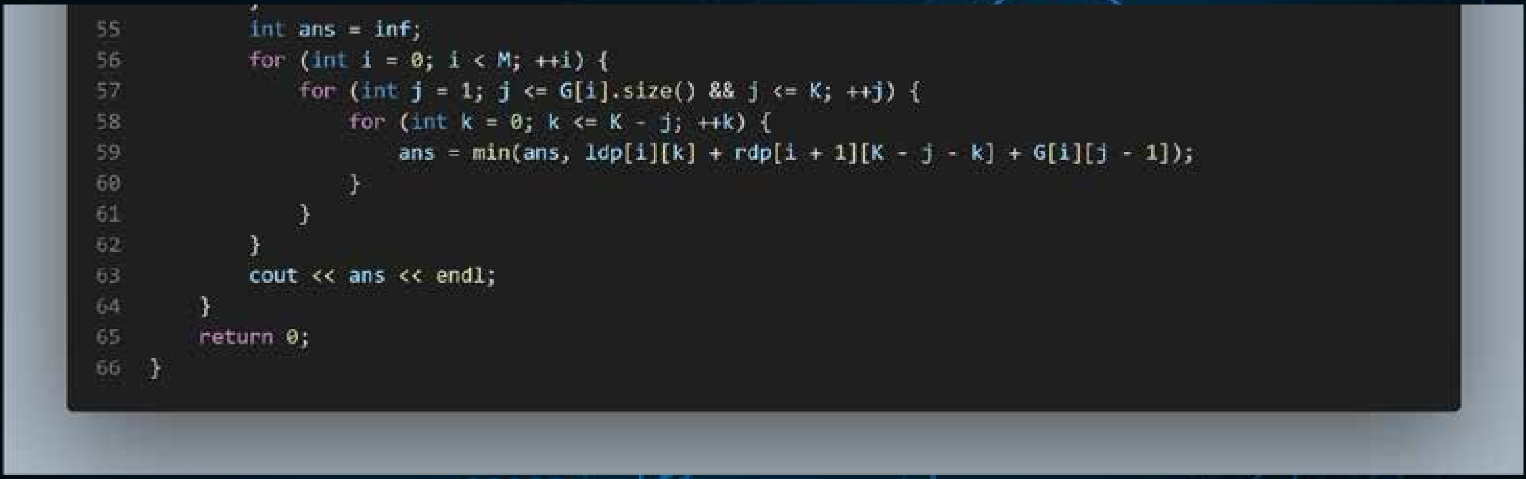
```
8
6
10028
```

SOLUTION



```cpp
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
const int inf = 1012345678;
int main() {
    cin.tie(0);
    ios_base::sync_with_stdio(false);
    int Q;
    cin >> Q;
    for (int rep = 1; rep <= Q; ++rep) {
        int N, K;
        cin >> N >> K;
        vector<int> P(N), A(N);
        for (int i = 0; i < N; ++i) cin >> P[i];
        for (int i = 0; i < N; ++i) cin >> A[i];
        vector<int> comp = A;
        sort(comp.begin(), comp.end());
        comp.erase(unique(comp.begin(), comp.end()), comp.end());
        for (int i = 0; i < N; ++i) {
            A[i] = lower_bound(comp.begin(), comp.end(), A[i]) - comp.begin();
        }
        int M = comp.size();
        vector<vector<int> > G(M);
        for (int i = 0; i < N; ++i) {
            G[A[i]].push_back(P[i]);
        }
        for (int i = 0; i < M; ++i) {
            sort(G[i].begin(), G[i].end());
        }
        vector<vector<int> > ldp(M + 1, vector<int>(K + 1, inf)), rdp(M + 1, vector<int>(K + 1, inf));
        ldp[0][0] = 0; rdp[M][0] = 0;
        for (int i = 0; i < M; ++i) {
            ldp[i + 1] = ldp[i];
            for (int j = 1; j <= G[i].size(); ++j) {
                for (int k = j; k <= K; ++k) {
                    ldp[i + 1][k] = min(ldp[i + 1][k], ldp[i][k - j] + G[i][j - 1] * 2);
                }
            }
            for (int j = K - 1; j >= 0; --j) {
                ldp[i + 1][j] = min(ldp[i + 1][j], ldp[i + 1][j + 1]);
            }
        }
        for (int i = M - 1; i >= 0; --i) {
            rdp[i] = rdp[i + 1];
            for (int j = 1; j <= G[i].size(); ++j) {
                for (int k = j; k <= K; ++k) {
                    rdp[i][k] = min(rdp[i][k], rdp[i + 1][k - j] + G[i][j - 1] * 2);
                }
            }
            for (int j = K - 1; j >= 0; --j) {
                rdp[i][j] = min(rdp[i][j], rdp[i][j + 1]);
            }
        }
        int ans = inf;
```

SOLUTION



```
        int ans = inf;
        for (int i = 0; i < M; ++i) {
            for (int j = 1; j <= G[i].size() && j <= K; ++j) {
                for (int k = 0; k <= K - j; ++k) {
                    ans = min(ans, ldp[i][k] + rdp[i + 1][K - j - k] + G[i][j - 1]);
                }
            }
        }
        cout << ans << endl;
    }
    return 0;
}
```