Problem

Bundle is an animal researcher and needs to go observe K dogs. She lives on a horizontal street marked at metre incr ments with consecutive numbers 0, 1, 2, 3 and so on. She begins in her home, which is at position 0. There are also dogs on the street. The i-th dog is Pi metres to the right of her home on the street (multiple dogs can share the same osition).

Dogs come in different colors, which are denoted by positive integers. The i-th animal is of color Ai.

If Bundle is at her home, she can change the current color of her shirt. This is important since the dogs are very shy! undle can only observe a dog if she is at the same position as that dog, and is wearing a shirt of the same color as the dog.

It takes Bundle one second to move one metre to the left or right on the street. It takes her no time to change shirts or observe a dog.

What is the least amount of time it will take Bundle to observe K dogs? Note that she does not have to return home a ter observing K dogs.

Input

The first line of the input gives the number of test cases, T. T test cases follow. Each testcase begins with a line cont ining the two integers N and K, the number of dogs on the number line and the number of dogs Bundle needs to obs rve, respectively. The second line contains N integers, the i-th of which is Pi, the position of the i-th dog. The third li e contains N integers, the i-th of which is Ai, the color of the i-th dog.

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is th least time Bundle needs to observe K dogs.

Limits

Time limit: 30 seconds per test set.
Memory limit: 1GB.
$1 \leq T \leq 100$.
$1 \leq K \leq N$.
$1 \leq Ai \leq 1000$.
$1 \leq Pi \leq 105$.

Test set 1 (Visible)

$1 \leq N \leq 50$.

Test set 2 (Hidden)

$1 \leq N \leq 1000$.

Sample

Input

Output

```
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

Case #1: 8
Case #2: 6
Case #3: 10028

In Sample Case #1, there are N = 4 dogs and Bundle needs to observe K = 3 dogs. One way that she can achieve this is as follows:
Put on a shirt of color 3.
Move one metre to the right and observe the dog there.
Move one metre to the right again and observe the dog there.
Move two metres to the left, returning to her home.
Change into a shirt of color 2.
Move four metres to the right and observe the dog there.
In total, this takes Bundle 8 seconds which is the least time possible, so the answer is 8.

In Sample Case #2, there are N = 4 dogs and Bundle needs to observe K = 3 dogs. One way that she can achieve this is as follows:
Put on a shirt of color 1.
Move one metre to the right and observe the dog there.
Move one metre to the left, returning to her home.
Change into a shirt of color 2.
Move two metres to the right and observe the dog there.
Move two metres to the right again and observe the dog there. Note that Bundle is unable to observe the dog she pas es at position 3, since her shirt is the wrong color (even though she was wearing the right colored shirt previously).
In total, this takes Bundle 6 seconds which is the least time possible, so the answer is 6.

In Sample Case #3, note that:
Multiple dogs can share the same position and
Dogs are not necessarily given in ascending order of position.
No explanation is provided for the answer to this case.

□ □
Syntax pre-check
Show Test Input

```
#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;
 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 << "Case #" << rep << ": " << ans << endl;
 }
 return 0;
}
```