COMPUTER SOCIETY OF INDIA (CSI)
COEP TECH STUDENT CHAPTER

PRESENTS

CODE QUEST 5.0

EDITIORIAL

BATTLE OF THE BOTS

PROBLEM STATEMENT

As part of the COEP Annual Tech Fest, the Robotics Club has designed two advanced robots, Astra and Zephyr, to face off in an exciting battle of skill and strategy.

The competition features N different energy modules, each designed with unique properties by the Robotics Club. These energy modules, numbered from 1 to N, are essential for powering the robots' advanced functionalities.

Before each battle, as the referee, you decide which types of energy modules will be made available for the fight. You will choose a range of modules, defined by two integers, L and R ($1 \leq L \leq R \leq N$). Only the energy modules from type L to type R (inclusive) will be accessible to the robots.

PROBLEM STATEMENT

Each robot is programmed with specific efficiency levels for each energy module type, represented by arrays Ai (for Astra) and Zi (for Zephyr). When the modules are made available, both Astra and Zephyr will select the module that gives them the highest efficiency. If a robot finds multiple modules with the same highest efficiency, it will choose one at random.

To keep the battle fair yet exciting, the maximum difference between Astra's and Zephyr's efficiency levels with their chosen energy modules must not exceed K.

Your task is to determine how many valid pairs (L, R) result in a fair battle.

INPUT FORMAT

1. The first line of the input contains the number of test cases, T.

2. For each test case:
   - The first line contains two integers, N (number of energy module types) and K (maximum allowed efficiency difference).
   - The second line contains N integers, Ai, representing Astra's efficiency levels for each energy module type.
   - The third line contains N integers, Zi, representing Zephyr's efficiency levels for each energy module type.

OUTPUT FORMAT & CONSTRAINTS

$1 \leq T \leq 100$.
$1 \leq N \leq 1000$.
$0 \leq K \leq 105$.
$0 \leq Ai \leq 10^5$, for all i.

For each test case, output a single line containing one integer, which is the maximum possible number of valid (L, R) pairs that result in a fair battle.

SAMPLE TESTCASES

SAMPLE INPUT :

```
6
4 0
1 1 1 8
8 8 8 8
3 0
0 1 1
1 1 0
1 0
3
3
5 0
0 8 0 8 0
4 0 4 0 4
3 0
1 0 0
0 1 2
5 2
1 2 3 4 5
5 5 5 5 10
```

SAMPLE TESTCASES

SAMPLE OUTPUT :

4
4
1
0
1
7

SOLUTION

```cpp
#include <bits/stdc++.h>
#define sz(x) ((int)x.size())
#define all(x) (x).begin(), (x).end()
using namespace std;
typedef long long ll;
typedef long double ld;
int T, n, K, a[100010], b[100010], s[100010];
int ta[100010], tb[100010];
ll solve(int l, int r) {
    if(l == r) return abs(a[l] - b[l]) <= K;
    int mid = (l + r) / 2;
    ll ret = solve(l, mid) + solve(mid + 1, r);
    ta[1] = a[mid+1];
    tb[1] = b[mid+1];
    s[1] = abs(ta[1] - tb[1]) <= K;
    for(int i=2; i<=r-mid; i++) {
        ta[i] = max(ta[i-1], a[mid+i]);
        tb[i] = max(tb[i-1], b[mid+i]);
        s[i] = s[i-1] + (abs(ta[i] - tb[i]) <= K);
    }
    int x = 0, y = 0;
    for(int i=mid, pa, pb; i>=l; i--) {
        x = max(x, a[i]);
        y = max(y, b[i]);
        pa = upper_bound(ta+1, ta+r-mid+1, x) - ta;
        pb = upper_bound(tb+1, tb+r-mid+1, y) - tb;
        if(abs(x - y) <= K) ret += min(pa, pb) - 1;
        if(pa < pb) {
            int i1 = lower_bound(ta+pa, ta+pb, y - K) - ta;
            int i2 = lower_bound(ta+pa, ta+pb, y + K + 1) - ta - 1;
            if(i1 <= i2) ret += i2 - i1 + 1;
        } else if(pa > pb) {
            int i1 = lower_bound(tb+pb, tb+pa, x - K) - tb;
            int i2 = lower_bound(tb+pb, tb+pa, x + K + 1) - tb - 1;
            if(i1 <= i2) ret += i2 - i1 + 1;
        }
        ret += s[r-mid] - s[max(pa, pb) - 1];
    }
    return ret;
}
int main() {
    scanf("%d", &T);
    for(int t=1; t<=T; t++) {
        scanf("%d%d", &n, &K);
        for(int i=1; i<=n; i++)
            scanf("%d", a+i);
        for(int i=1; i<=n; i++)
            scanf("%d", b+i);
        for(int i=1; i<=n; i++)
            s[i] = s[i-1] + (abs(a[i] - b[i]) <= K);
        printf("%lld\n",solve(1, n));
    }
    return 0;
}
```