Problem

En garde! Charles and Delila are about to face off against each other in the final fight of the Swordmaster fencing to rnament.

Along one wall of the fencing arena, there is a rack with N different types of swords; the swords are numbered by ty e, from 1 to N. As the head judge, you will pick a pair of integers (L, R) (with $1 \leq L \leq R \leq N$), and only the L-th thro gh R-th types of swords (inclusive) will be available for the fight.

Different types of sword are used in different ways, and being good with one type of sword does not necessarily mea you are good with another! Charles and Delila have skill levels of Ci and Di, respectively, with the i-th type of swor . Each of them will look at the types of sword you have made available for this fight, and then each will choose a typ with which they are most skilled. If there are multiple available types with which a fighter is equally skilled, and tha skill level exceeds the fighter's skill level in all other available types, then the fighter will make one of those equally good choices at random. Notice that it is possible for Charles and Delila to choose the same type of sword, which is ine — there are multiple copies of each type of sword available.

The fight is fair if the absolute difference between Charles's skill level with his chosen sword type and Delila's skill l vel with her chosen sword type is at most K. To keep the fight exciting, you'd like to know how many different pairs (L, R) you can choose that will result in a fair fight.

Input

The first line of the input gives the number of test cases, T. T test cases follow. Each case begins with a line containi g the two integers N and K, as described above. Then, two more lines follow. The first of these lines contains N inte ers Ci, giving Charles' skill levels for each type of sword, as described above. Similarly, the second line contains N i tegers Di, giving Delila's skill levels.

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 number of choices you can make that result in a fair fight, as described above.

Limits

$1 \leq T \leq 100$.
$0 \leq K \leq 105$.
$0 \leq Ci \leq 105$, for all i.
$0 \leq Di \leq 105$, for all i.
Time limit: 30 seconds per test set.
Memory limit: 1GB.

Test set 1 (Visible)

$1 \leq N \leq 100$.

Test set 2 (Hidden)

$N = 105$, for exactly 8 test cases.
$1 \leq N \leq 1000$, for all but 8 test cases.

Sample

Input

Output

```
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
```

```
Case #1: 4
Case #2: 4
Case #3: 1
Case #4: 0
Case #5: 1
Case #6: 7
```

In Sample Case #1, the fight is fair if and only if Charles can use the last type of sword, so the answer is 4.

In Sample Case #2, there are 4 fair fights: (1, 2), (1, 3), (2, 2), and (2, 3). Notice that for pairs like (1, 3), Charles and Delila both have multiple swords they could choose that they are most skilled with; however, each pair only counts a one fair fight.

In Sample Case #3, there is 1 fair fight: (1, 1).

In Sample Case #4, there are no fair fights, so the answer is 0.

In Sample Case #5, remember that the duelists are not trying to make the fights fair; they choose the type of sword t at they are most skilled with. For example, (1, 3) is not a fair fight, because Charles will choose the first type of swo d, and Delila will choose the third type of sword. Delila will not go easy on Charles by choosing a weaker sword!

In Sample Case #6, there are 7 fair fights: (1, 3), (1, 4), (2, 3), (2, 4), (3, 3), (3, 4), and (4, 4).

Solution:

```
#pragma GCC optimize ("O3")
#pragma GCC target ("sse4")

#include <bits/stdc++.h>
```

```
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/rope>

using namespace std;
using namespace __gnu_pbds;
using namespace __gnu_cxx;

typedef long long ll;
typedef long double ld;
typedef complex<ld> cd;

typedef pair<int, int> pi;
typedef pair<ll,ll> pl;
typedef pair<ld,ld> pd;

typedef vector<int> vi;
typedef vector<ld> vd;
typedef vector<ll> vl;
typedef vector<pi> vpi;
typedef vector<pl> vpl;
typedef vector<cd> vcd;

template <class T> using Tree = tree<T, null_type, less<T>, rb_tree_tag,tree_order_statistics_node_update>;

#define FOR(i, a, b) for (int i = (a); i < (b); i++)
#define F0R(i, a) for (int i = 0; i < (a); i++)
#define FORd(i,a,b) for (int i = (b)-1; i >= (a); i--)
#define F0Rd(i,a) for (int i = (a)-1; i >= 0; i--)
#define trav(a, x) for (auto& a : x)

#define mp make_pair
#define pb push_back
#define f first
#define s second
#define lb lower_bound
#define ub upper_bound

#define sz(x) (int)x.size()
#define beg(x) x.begin()
#define en(x) x.end()
#define all(x) beg(x), en(x)
#define resz resize

const int MOD = 1000000007; // 998244353
const ll INF = 1e18;
const int MX = 100005;
const ld PI = 4*atan((ld)1);

template<class T> void ckmin(T &a, T b) { a = min(a, b); }
template<class T> void ckmax(T &a, T b) { a = max(a, b); }
```

```
namespace input {
   template<class T> void re(complex<T>& x);
   template<class T1, class T2> void re(pair<T1,T2>& p);
   template<class T> void re(vector<T>& a);
   template<class T, size_t SZ> void re(array<T,SZ>& a);

   template<class T> void re(T& x) { cin >> x; }
   void re(double& x) { string t; re(t); x = stod(t); }
   void re(ld& x) { string t; re(t); x = stold(t); }
   template<class Arg, class... Args> void re(Arg& first, Args&... rest) {
      re(first); re(rest...);
   }

   template<class T> void re(complex<T>& x) { T a,b; re(a,b); x = cd(a,b); }
   template<class T1, class T2> void re(pair<T1,T2>& p) { re(p.f,p.s); }
   template<class T> void re(vector<T>& a) { F0R(i,sz(a)) re(a[i]); }
   template<class T, size_t SZ> void re(array<T,SZ>& a) { F0R(i,SZ) re(a[i]); }
}

using namespace input;

namespace output {
   template<class T1, class T2> void pr(const pair<T1,T2>& x);
   template<class T, size_t SZ> void pr(const array<T,SZ>& x);
   template<class T> void pr(const vector<T>& x);
   template<class T> void pr(const set<T>& x);
   template<class T1, class T2> void pr(const map<T1,T2>& x);

   template<class T> void pr(const T& x) { cout << x; }
   template<class Arg, class... Args> void pr(const Arg& first, const Args&... rest) {
      pr(first); pr(rest...);
   }

   template<class T1, class T2> void pr(const pair<T1,T2>& x) {
      pr("{",x.f,", ",x.s,"}");
   }
   template<class T> void prContain(const T& x) {
      pr("{");
      bool fst = 1; trav(a,x) pr(!fst?", ":"",a), fst = 0;
      pr("}");
   }
   template<class T, size_t SZ> void pr(const array<T,SZ>& x) { prContain(x); }
   template<class T> void pr(const vector<T>& x) { prContain(x); }
   template<class T> void pr(const set<T>& x) { prContain(x); }
   template<class T1, class T2> void pr(const map<T1,T2>& x) { prContain(x); }

   void ps() { pr("\n"); }
   template<class Arg, class... Args> void ps(const Arg& first, const Args&... rest) {
      pr(first," "); ps(rest...); // print w/ spaces
   }
}
```

```
using namespace output;

namespace io {
   void setIn(string s) { freopen(s.c_str(),"r",stdin); }
   void setOut(string s) { freopen(s.c_str(),"w",stdout); }
   void setIO(string s = "") {
      ios_base::sync_with_stdio(0); cin.tie(0); // fast I/O
      if (sz(s)) { setIn(s+".in"), setOut(s+".out"); } // for USACO
   }
}

using namespace io;

template<class T> T invGeneral(T a, T b) {
   a %= b; if (a == 0) return b == 1 ? 0 : -1;
   T x = invGeneral(b,a);
   return x == -1 ? -1 : ((1-(ll)b*x)/a+b)%b;
}

template<class T> struct modular {
   T val;
   explicit operator T() const { return val; }
   modular() { val = 0; }
   template<class U> modular(const U& v) {
      val = (-MOD <= v && v <= MOD) ? v : v % MOD;
      if (val < 0) val += MOD;
   }
   friend ostream& operator<<(ostream& os, const modular& a) { return os << a.val; }
   friend bool operator==(const modular& a, const modular& b) { return a.val == b.val; }
   friend bool operator!=(const modular& a, const modular& b) { return !(a == b); }

   modular operator-() const { return modular(-val); }
   modular& operator+=(const modular& m) { if ((val += m.val) >= MOD) val -= MOD; return *this; }
   modular& operator-=(const modular& m) { if ((val -= m.val) < 0) val += MOD; return *this; }
   modular& operator*=(const modular& m) { val = (ll)val*m.val%MOD; return *this; }
   friend modular exp(modular a, ll p) {
      modular ans = 1; for (; p; p /= 2, a *= a) if (p&1) ans *= a;
      return ans;
   }
   friend modular inv(const modular& a) { return invGeneral(a.val,MOD); }
   // inv is equivalent to return exp(b,b.mod-2) if prime
   modular& operator/=(const modular& m) { return (*this) *= inv(m); }

   friend modular operator+(modular a, const modular& b) { return a += b; }
   friend modular operator-(modular a, const modular& b) { return a -= b; }
   friend modular operator*(modular a, const modular& b) { return a *= b; }

   friend modular operator/(modular a, const modular& b) { return a /= b; }
};

typedef modular<int> mi;
typedef pair<mi,mi> pmi;
```

```
typedef vector<mi> vmi;
typedef vector<pmi> vpmi;

template<class T, int SZ> struct RMQ {
  T stor[SZ][32-__builtin_clz(SZ)];
  vi x;

  T comb(T a, T b) {
    if (x[a] > x[b]) return a;
    return b;
  }

  void build(vi _x) {
    x = _x;
    F0R(i,sz(x)) stor[i][0] = i;
    FOR(j,1,32-__builtin_clz(SZ)) F0R(i,SZ-(1<<(j-1)))
      stor[i][j] = comb(stor[i][j-1],
          stor[i+(1<<(j-1))][j-1]);
  }

  T query(int l, int r) {
    int x = 31-__builtin_clz(r-l+1);
    return comb(stor[l][x],stor[r-(1<<x)+1][x]);
  }
};

RMQ<int,MX> R[2];
int N,K;
vi C,D;
ll ans;

ll getRange(int l, int m, int r, int x) { // everything <= x
  if (D[m] > x) return 0;
  int lo = m, hi = r;
  while (lo < hi) {
    int mid = (lo+hi+1)/2;
    if (D[R[1].query(m,mid)] <= x) lo = mid;
    else hi = mid-1;
  }
  int RR = hi;
  lo = l, hi = m;
  while (lo < hi) {
    int mid = (lo+hi)/2;
    if (D[R[1].query(mid,m)] <= x) hi = mid;
    else lo = mid+1;
  }
  int LL = hi;
  return (ll)(RR-m+1)*(m-LL+1);
}

void divi(int l, int r) {
  if (l > r) return;
```

```
    int m = R[0].query(l,r);
    divi(l,m-1); divi(m+1,r);
    ans += getRange(l,m,r,C[m]+K);
    ans -= getRange(l,m,r,C[m]-K-1);
}

void solve(int caseNum) {
    re(N,K); C.resz(N), D.resz(N); re(C,D);
    R[0].build(C), R[1].build(D);
    ans = 0;
    divi(0,N-1);
    ps(ans);
    // cerr << "Solved #" << caseNum << "\n";
}

int main() {
    setIO();
    int T; re(T);
    FOR(i,1,T+1) {
        pr("Case #",i,": ");
        solve(i);
    }
}

/* stuff you should look for
    * int overflow, array bounds
    * special cases (n=1?), set tle
    * do smth instead of nothing and stay organized
*/
```