Permuting Two Arrays

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

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);

/*
 * Complete the 'twoArrays' function below.
 *
 * The function is expected to return a STRING.
 * The function accepts following parameters:
 *  1. INTEGER k
 *  2. INTEGER_ARRAY A
 *  3. INTEGER_ARRAY B
 */

string twoArrays(int k, vector<int> A, vector<int> B) {
    sort(A.begin(), A.end());
    sort(B.begin(), B.end(), greater<int>());
    for(int i=0;i<A.size();i++){
        if(A[i]+B[i]<k){
            return "NO";
        }
    }
    return "YES";
}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    string q_temp;
    getline(cin, q_temp);

    int q = stoi(ltrim(rtrim(q_temp)));

    for (int q_itr = 0; q_itr < q; q_itr++) {
        string first_multiple_input_temp;
        getline(cin, first_multiple_input_temp);
```

```
        vector<string> first_multiple_input =
split(rtrim(first_multiple_input_temp));

        int n = stoi(first_multiple_input[0]);

        int k = stoi(first_multiple_input[1]);

        string A_temp_temp;
        getline(cin, A_temp_temp);

        vector<string> A_temp = split(rtrim(A_temp_temp));

        vector<int> A(n);

        for (int i = 0; i < n; i++) {
            int A_item = stoi(A_temp[i]);

            A[i] = A_item;
        }

        string B_temp_temp;
        getline(cin, B_temp_temp);

        vector<string> B_temp = split(rtrim(B_temp_temp));

        vector<int> B(n);

        for (int i = 0; i < n; i++) {
            int B_item = stoi(B_temp[i]);

            B[i] = B_item;
        }

        string result = twoArrays(k, A, B);

        fout << result << "\n";
    }

    fout.close();

    return 0;
}

string ltrim(const string &str) {
    string s(str);
```

```
    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int,
int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int,
int>(isspace))).base(),
        s.end()
    );

    return s;
}

vector<string> split(const string &str) {
    vector<string> tokens;

    string::size_type start = 0;
    string::size_type end = 0;

    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push_back(str.substr(start, end - start));

        start = end + 1;
    }

    tokens.push_back(str.substr(start));

    return tokens;
}
```