#include <assert.h>

#include <ctype.h>

#include <limits.h>

#include <math.h>

#include <stdbool.h>

#include <stddef.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* readline();

char\* ltrim(char\*);

char\* rtrim(char\*);

char\*\* split\_string(char\*);

int parse\_int(char\*);

/\*

\* 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

\*/

int compare\_asc(const void \*a, const void \*b) {

return (\*(int\*)a - \*(int\*)b);

}

int compare\_desc(const void \*a, const void \*b) {

return (\*(int\*)b - \*(int\*)a);

}

char\* twoArrays(int k, int A\_count, int\* A, int B\_count, int\* B) {

qsort(A, A\_count, sizeof(int), compare\_asc);

qsort(B, B\_count, sizeof(int), compare\_desc);

for (int i = 0; i < A\_count; i++) {

if (A[i] + B[i] < k) {

return "NO";

}

}

return "YES";

}

int main() {

int q;

scanf("%d", &q);

while (q--) {

int n, k;

scanf("%d %d", &n, &k);

int\* A = (int\*)malloc(n \* sizeof(int));

int\* B = (int\*)malloc(n \* sizeof(int));

for (int i = 0; i < n; i++) {

scanf("%d", &A[i]);

}

for (int i = 0; i < n; i++) {

scanf("%d", &B[i]);

}

char\* result = twoArrays(k, n, A, n, B);

printf("%s\n", result);

free(A);

free(B);

}

return 0;

}

char\* readline() {

size\_t alloc\_length = 1024;

size\_t data\_length = 0;

char\* data = malloc(alloc\_length);

while (true) {

char\* cursor = data + data\_length;

char\* line = fgets(cursor, alloc\_length - data\_length, stdin);

if (!line) {

break;

}

data\_length += strlen(cursor);

if (data\_length < alloc\_length - 1 || data[data\_length - 1] == '\n') {

break;

}

alloc\_length <<= 1;

data = realloc(data, alloc\_length);

if (!data) {

data = '\0';

break;

}

}

if (data[data\_length - 1] == '\n') {

data[data\_length - 1] = '\0';

data = realloc(data, data\_length);

if (!data) {

data = '\0';

}

} else {

data = realloc(data, data\_length + 1);

if (!data) {

data = '\0';

} else {

data[data\_length] = '\0';

}

}

return data;

}

char\* ltrim(char\* str) {

if (!str) {

return '\0';

}

if (!\*str) {

return str;

}

while (\*str != '\0' && isspace(\*str)) {

str++;

}

return str;

}

char\* rtrim(char\* str) {

if (!str) {

return '\0';

}

if (!\*str) {

return str;

}

char\* end = str + strlen(str) - 1;

while (end >= str && isspace(\*end)) {

end--;

}

\*(end + 1) = '\0';

return str;

}

char\*\* split\_string(char\* str) {

char\*\* splits = NULL;

char\* token = strtok(str, " ");

int spaces = 0;

while (token) {

splits = realloc(splits, sizeof(char\*) \* ++spaces);

if (!splits) {

return splits;

}

splits[spaces - 1] = token;

token = strtok(NULL, " ");

}

return splits;

}

int parse\_int(char\* str) {

char\* endptr;

int value = strtol(str, &endptr, 10);

if (endptr == str || \*endptr != '\0') {

exit(EXIT\_FAILURE);

}

return value;

}