#include <assert.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\*\* split\_string(char\*);

// Complete the angryProfessor function below.

// Please either make the string static or allocate on the heap. For example,

// static char str[] = "hello world";

// return str;

//

// OR

//

// char\* str = "hello world";

// return str;

//

char\* angryProfessor(int k, int a\_count, int\* a) {

int count;

count =0;

int i;

for(i=0;i<a\_count;i++)

{

if(a[i]==0 || a[i]<0)

{

count++;

}

}

if(count>=k)

{

char \*str1 = "NO";

return str1;

}

else{

char \*str1 = "YES";

return str1;

}

}

int main()

{

FILE\* fptr = fopen(getenv("OUTPUT\_PATH"), "w");

char\* t\_endptr;

char\* t\_str = readline();

int t = strtol(t\_str, &t\_endptr, 10);

if (t\_endptr == t\_str || \*t\_endptr != '\0') { exit(EXIT\_FAILURE); }

for (int t\_itr = 0; t\_itr < t; t\_itr++) {

char\*\* nk = split\_string(readline());

char\* n\_endptr;

char\* n\_str = nk[0];

int n = strtol(n\_str, &n\_endptr, 10);

if (n\_endptr == n\_str || \*n\_endptr != '\0') { exit(EXIT\_FAILURE); }

char\* k\_endptr;

char\* k\_str = nk[1];

int k = strtol(k\_str, &k\_endptr, 10);

if (k\_endptr == k\_str || \*k\_endptr != '\0') { exit(EXIT\_FAILURE); }

char\*\* a\_temp = split\_string(readline());

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

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

char\* a\_item\_endptr;

char\* a\_item\_str = \*(a\_temp + i);

int a\_item = strtol(a\_item\_str, &a\_item\_endptr, 10);

if (a\_item\_endptr == a\_item\_str || \*a\_item\_endptr != '\0') { exit(EXIT\_FAILURE); }

\*(a + i) = a\_item;

}

int a\_count = n;

char\* result = angryProfessor(k, a\_count, a);

fprintf(fptr, "%s\n", result);

}

fclose(fptr);

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; }

size\_t new\_length = alloc\_length << 1;

data = realloc(data, new\_length);

if (!data) { break; }

alloc\_length = new\_length;

}

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

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

}

data = realloc(data, data\_length);

return data;

}

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;

}