#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\*);

// Complete the compareTriplets function below.

/\*

\* To return the integer array from the function, you should:

\* - Store the size of the array to be returned in the result\_count variable

\* - Allocate the array statically or dynamically

\*

\* For example,

\* int\* return\_integer\_array\_using\_static\_allocation(int\* result\_count) {

\* \*result\_count = 5;

\*

\* static int a[5] = {1, 2, 3, 4, 5};

\*

\* return a;

\* }

\*

\* int\* return\_integer\_array\_using\_dynamic\_allocation(int\* result\_count) {

\* \*result\_count = 5;

\*

\* int \*a = malloc(5 \* sizeof(int));

\*

\* for (int i = 0; i < 5; i++) {

\* \*(a + i) = i + 1;

\* }

\*

\* return a;

\* }

\*

\*/int ans[2];

int\* compareTriplets(int a\_count, int\* a, int b\_count, int\* b, int\* result\_count) {

int alice,bob;

alice=0;

bob=0;

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

{

if(a[i]>b[i])

{

alice++;

}

else if(a[i]==b[i])

{

}else{

bob++;

}

}

ans[0] = alice;

ans[1] = bob;

\*result\_count = 2;

return ans;

}

int main()

{

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

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

int\* a = malloc(3 \* sizeof(int));

for (int i = 0; i < 3; 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 = 3;

char\*\* b\_temp = split\_string(rtrim(readline()));

int\* b = malloc(3 \* sizeof(int));

for (int i = 0; i < 3; i++) {

char\* b\_item\_endptr;

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

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

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

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

}

int b\_count = 3;

int result\_count;

int\* result = compareTriplets(a\_count, a, b\_count, b, &result\_count);

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

fprintf(fptr, "%d", \*(result + i));

if (i != result\_count - 1) {

fprintf(fptr, " ");

}

}

fprintf(fptr, "\n");

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;

}

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;

}