#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 breakingRecords function below.

// Please store the size of the integer array to be returned in result\_count pointer. For example,

// int a[3] = {1, 2, 3};

//

// \*result\_count = 3;

//

// return a;

//

int ans[2] = {0,0};

int\* breakingRecords(int scores\_count, int\* scores, int\* result\_count) {

int min,max;

min = scores[0];

max = scores[0];

int low,high;

int i;

high =0;

low =0;

for(i=1;i<scores\_count;i++)

{

if(scores[i]>max)

{

max = scores[i];

high++;

}

if(scores[i]<min)

{

min = scores[i];

low++;

}

}

ans[0] = high;

ans[1] = low;

\*result\_count = 2;

return ans;

}

int main()

{

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

char\* n\_endptr;

char\* n\_str = readline();

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

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

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

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

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

char\* scores\_item\_endptr;

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

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

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

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

}

int scores\_count = n;

int result\_count;

int\* result = breakingRecords(scores\_count, scores, &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 (!line) {

break;

}

}

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

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

data = realloc(data, data\_length);

} else {

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

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

}

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;

}