#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 birthday function below.

int birthday(int s\_count, int\* s, int d, int m) {

int i;

int j;

int sum;

int counter;

sum =0;

counter =0;

if(s\_count ==1 && s[0]==d)

{

return 1;

}

for(i=0;i<s\_count-m+1;i++)

{

sum =0;

for(j=0;j<m;j++)

{

sum = sum + s[i+j];

}

if(sum == d)

{

counter++;

}

}

return counter;

}

int main()

{

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

char\* n\_endptr;

char\* n\_str = ltrim(rtrim(readline()));

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

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

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

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

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

char\* s\_item\_endptr;

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

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

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

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

}

int s\_count = n;

char\*\* dm = split\_string(rtrim(readline()));

char\* d\_endptr;

char\* d\_str = dm[0];

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

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

char\* m\_endptr;

char\* m\_str = dm[1];

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

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

int result = birthday(s\_count, s, d, m);

fprintf(fptr, "%d\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;

}

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;

}