#include <assert.h>

#include <limits.h>

#include <math.h>

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* readline();

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

/\*

\* Complete the getMoneySpent function below.

\*/

int getMoneySpent(int keyboards\_count, int\* keyboards, int drives\_count, int\* drives, int b) {

int i,j,max;

max = -1;

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

{

for(j=0;j<drives\_count;j++)

{

if((keyboards[i] + drives[j]) > max && (keyboards[i]+drives[j])<=b)

{

max = keyboards[i] + drives[j];

}

}

}

return max;

}

int main()

{

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

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

char\* b\_endptr;

char\* b\_str = bnm[0];

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

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

char\* n\_endptr;

char\* n\_str = bnm[1];

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

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

char\* m\_endptr;

char\* m\_str = bnm[2];

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

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

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

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

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

char\* keyboards\_item\_endptr;

char\* keyboards\_item\_str = \*(keyboards\_temp + keyboards\_itr);

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

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

\*(keyboards + keyboards\_itr) = keyboards\_item;

}

int keyboards\_count = n;

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

int\* drives = malloc(m \* sizeof(int));

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

char\* drives\_item\_endptr;

char\* drives\_item\_str = \*(drives\_temp + drives\_itr);

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

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

\*(drives + drives\_itr) = drives\_item;

}

int drives\_count = m;

/\*

\* The maximum amount of money she can spend on a keyboard and USB drive, or -1 if she can't purchase both items

\*/

int moneySpent = getMoneySpent(keyboards\_count, keyboards, drives\_count, drives, b);

fprintf(fptr, "%d\n", moneySpent);

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;

}