#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\*);

int parse\_int(char\*);

/\*

 \* Complete the 'equalStacks' function below.

 \*

 \* The function is expected to return an INTEGER.

 \* The function accepts following parameters:

 \*  1. INTEGER\_ARRAY h1

 \*  2. INTEGER\_ARRAY h2

 \*  3. INTEGER\_ARRAY h3

 \*/

int equalStacks(int n1, int\* h1, int n2, int\* h2, int n3, int\* h3) {

int i1=0,i2=0,i3=0;

int s1=0,s2=0,s3=0;

for(int i=0;i<n1;i++) s1+=h1[i];

for(int i=0;i<n2;i++) s2+=h2[i];

for(int i=0;i<n3;i++) s3+=h3[i];

while(1){

    if(i1==n1||i2==n2||i3==n3) return 0;

    if(s1==s2 && s2==s3) return s1;

    if(s1>=s2 && s1>=s3){

        s1-=h1[i1++];

    }else if (s2>=s1 && s2>=s3){

        s2-=h2[i2++];

    }else{

        s3-=h3[i3++];

    }

}

}

int main()

{

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

    char\*\* first\_multiple\_input = split\_string(rtrim(readline()));

    int n1 = parse\_int(\*(first\_multiple\_input + 0));

    int n2 = parse\_int(\*(first\_multiple\_input + 1));

    int n3 = parse\_int(\*(first\_multiple\_input + 2));

    char\*\* h1\_temp = split\_string(rtrim(readline()));

    int\* h1 = malloc(n1 \* sizeof(int));

    for (int i = 0; i < n1; i++) {

        int h1\_item = parse\_int(\*(h1\_temp + i));

        \*(h1 + i) = h1\_item;

    }

    char\*\* h2\_temp = split\_string(rtrim(readline()));

    int\* h2 = malloc(n2 \* sizeof(int));

    for (int i = 0; i < n2; i++) {

        int h2\_item = parse\_int(\*(h2\_temp + i));

        \*(h2 + i) = h2\_item;

    }

    char\*\* h3\_temp = split\_string(rtrim(readline()));

    int\* h3 = malloc(n3 \* sizeof(int));

    for (int i = 0; i < n3; i++) {

        int h3\_item = parse\_int(\*(h3\_temp + i));

        \*(h3 + i) = h3\_item;

    }

    int result = equalStacks(n1, h1, n2, h2, n3, h3);

    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;

}

int parse\_int(char\* str) {

    char\* endptr;

    int value = strtol(str, &endptr, 10);

    if (endptr == str || \*endptr != '\0') {

        exit(EXIT\_FAILURE);

    }

    return value;

}