Equal Stacks

```
#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);

/*
 * 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(vector<int> h1, vector<int> h2, vector<int> h3) {
    int sum1 = accumulate(h1.begin(), h1.end(), 0);
    int sum2 = accumulate(h2.begin(), h2.end(), 0);
    int sum3 = accumulate(h3.begin(), h3.end(), 0);

    int i = 0, j = 0, k = 0;

    while (true) {
        if (i == h1.size() || j == h2.size() || k == h3.size())
            return 0;

        if (sum1 == sum2 && sum2 == sum3)
            return sum1;

        if (sum1 >= sum2 && sum1 >= sum3) {
            sum1 -= h1[i];
            i++;
        }
        else if (sum2 >= sum1 && sum2 >= sum3) {
            sum2 -= h2[j];
            j++;
        }
        else {
            sum3 -= h3[k];
            k++;
```

```
        }
    }
}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    string first_multiple_input_temp;
    getline(cin, first_multiple_input_temp);

    vector<string> first_multiple_input =
split(rtrim(first_multiple_input_temp));

    int n1 = stoi(first_multiple_input[0]);

    int n2 = stoi(first_multiple_input[1]);

    int n3 = stoi(first_multiple_input[2]);

    string h1_temp_temp;
    getline(cin, h1_temp_temp);

    vector<string> h1_temp = split(rtrim(h1_temp_temp));

    vector<int> h1(n1);

    for (int i = 0; i < n1; i++) {
        int h1_item = stoi(h1_temp[i]);

        h1[i] = h1_item;
    }

    string h2_temp_temp;
    getline(cin, h2_temp_temp);

    vector<string> h2 temp = split(rtrim(h2 temp temp));

    vector<int> h2(n2);

    for (int i = 0; i < n2; i++) {
        int h2_item = stoi(h2_temp[i]);

        h2[i] = h2_item;
    }
```

```
    string h3_temp_temp;
    getline(cin, h3_temp_temp);

    vector<string> h3_temp = split(rtrim(h3_temp_temp));

    vector<int> h3(n3);

    for (int i = 0; i < n3; i++) {
        int h3_item = stoi(h3_temp[i]);

        h3[i] = h3_item;
    }

    int result = equalStacks(h1, h2, h3);

    fout << result << "\n";

    fout.close();

    return 0;
}

string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int,
int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int,
int>(isspace))).base(),
        s.end()
    );

    return s;
}
```

```
vector<string> split(const string &str) {
    vector<string> tokens;

    string::size_type start = 0;
    string::size_type end = 0;

    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push_back(str.substr(start, end - start));

        start = end + 1;
    }

    tokens.push_back(str.substr(start));

    return tokens;
}
```