Experiment Title : 1

Student Name: Rajiv Paul **UID: 20BCS1812**
Branch: CSE **Section/Group: 702 A**
Semester: 5 **Date of Performance:18/08/22**
Subject Name: Competitive Coding-I **Subject Code: 20CSP-314**

Problem Statement 1.1 Down to Zero II

You are given Q queries. Each query consists of a single number N. You can perform any of the 2 operations on N in each move:

1: If we take 2 integers a and b where $N = a \times b (a \neq 1, b \neq 1)$, then we can change $N = max(a, b)$

2: Decrease the value of N by 1.

Determine the minimum number of moves required to reduce the value of N to 0.

Input Format

The first line contains the integer Q.
The next Q lines each contain an integer, N.

Constraints

$1 \leq Q \leq 10^3$
$0 \leq N \leq 10^6$

Output Format

Output Q lines. Each line containing the minimum number of moves required to reduce the value of N to 0.

Sample Input

```
2
3
4
```

Sample Output

```
3
3
```

Explanation

For test case 1, We only have one option that gives the minimum number of moves.

Follow 3 -> 2 -> 1 -> 0. Hence, 3 moves.

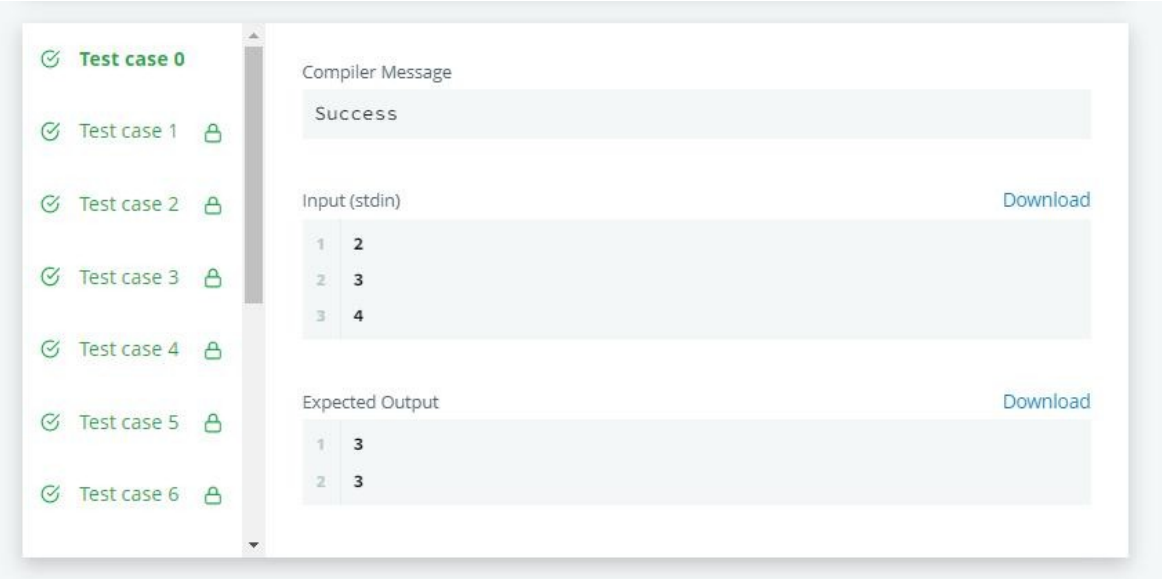
For the case 2, we can either go 4 -> 3 -> 2 -> 1 -> 0 or 4 -> 2 -> 1 -> 0. The 2nd option is more optimal. Hence, 3 moves.

Solution:

```
#include
<stdio.h>
#include
<math.h>

#define MAX    1000000
int

map[MAX+1];

int

main(void)

{
    int numCases =
    0;int i      =
    0;
    int n        = 0;
    int sqrt_max = sqrt(MAX);

    for (int i=0; i <= MAX;
        i++)map[i] = i;

    for (int i=1; i < MAX; i+
        +) {int score = map[i]
```

```
    + 1; int limit;
        if (map[i+1] >
            score)map[i+1]
            = score;

        if (i >
            sqrt_max)
            limit = MAX;
        else
            limit = i*i;
        for (int j = i+i; j <= limit; j +=
            i) {if (map[j] > score)
                map[j] = score;
        }
    }

    if (scanf("%d", &numCases) !=
        1)return 1;
    while (numCases-- > 0)
        {scanf("%d", &n);
        printf("%d\n", map[n]);
    }
    return 0;
}
```

Output:



Problem Statement 1.2 Truck Tour

Suppose there is a circle. There are N petrol pumps on that circle. Petrol pumps are numbered 0 to $(N-1)$ (both inclusive). You have two pieces of information corresponding to each of the petrol pump: (1) the amount of petrol that particular petrol pump will give, and (2) the distance from that petrol pump to the next petrol pump.

Initially, you have a tank of infinite capacity carrying no petrol. You can start the tour at any of the petrol pumps. Calculate the first point from where the truck will be able to complete the circle. Consider that the truck will stop at each of the petrol pumps. The truck will move one kilometer for each litre of the petrol.

Input Format

The first line will contain the value of N.

The next N lines will contain a pair of integers each, i.e. the amount of petrol that petrol pump will give and the distance between that petrol pump and the next petrol pump.

Constraints:

$1 \leq N \leq 10^5$

$1 \leq \text{amount of petrol, distance} \leq 10^9$

Output Format

An integer which will be the smallest index of the petrol pump from which we can start the tour.

Sample Input

```
3
1 5
10 3
3 4
```

Sample Output

```
1
```

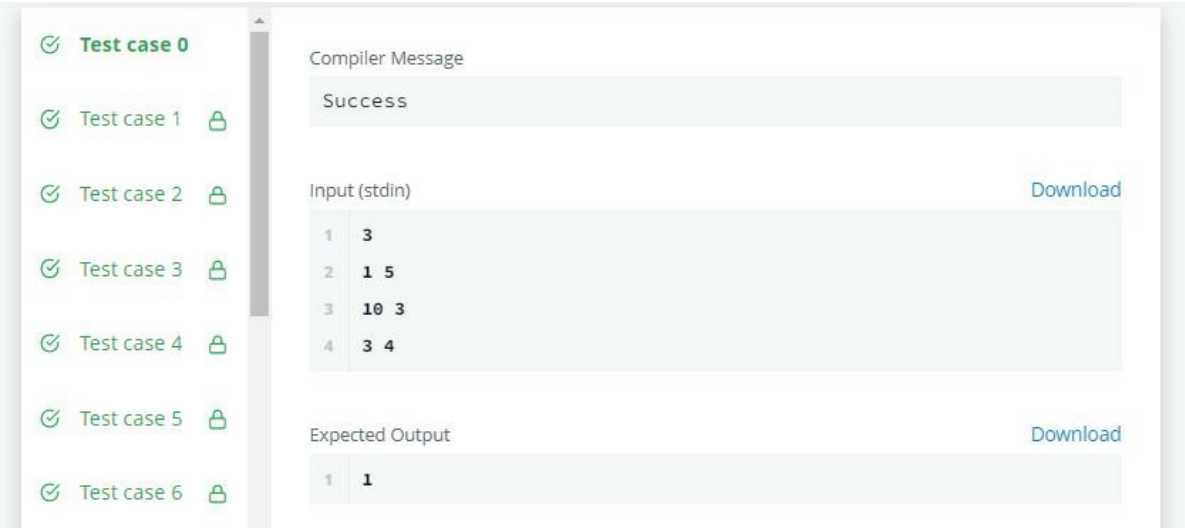
Explanation

We can start the tour from the second petrol pump.

Solution:

```
#include <cmath>
#include <cstdio>
#include <vector>
#include
<iostream>
#include
<algorithm>using
namespace std;

int n, p[100005], d[100005];
int main() {
    scanf("%d", &n);
    for (int i = 0; i < n; ++i) scanf("%d%d", &p[i],
    &d[i]);int ret = 0, amount = 0, sum = 0;
    for (int i = 0; i < n; +
        +i) {p[i] -= d[i];
        sum += p[i];
        if (amount + p[i] < 0)
            {amount = 0;
            ret = i + 1;
        } else amount += p[i];
    }
    printf("%d\n", sum >= 0 ? ret :
    -1);return 0;
}
```

Output:



Problem Statement 2.1 compare-the-triplets

Alice and Bob each created one problem for HackerRank. A reviewer rates the two challenges, awarding points on a scale from 1 to 100 for three categories: problem clarity, originality, and difficulty.

The rating for Alice's challenge is the triplet a = (a[0], a[1], a[2]), and the rating for Bob's challenge is the triplet b = (b[0], b[1], b[2]).

The task is to find their comparison points by comparing a[0] with b[0], a[1] with b[1], and a[2] with b[2].

- If a[i] > b[i], then Alice is awarded 1 point.
- If a[i] < b[i], then Bob is awarded 1 point.
- If a[i] = b[i], then neither person receives a point.

Comparison points is the total points a person earned.

Given a and b, determine their respective comparison points.

Example

a = [1, 2, 3]

b = [3, 2, 1]

- For elements *0*, Bob is awarded a point because a[0] .
- For the equal elements a[1] and b[1], no points are earned.
- Finally, for elements 2, a[2] > b[2] so Alice receives a point.

The return array is [1, 1] with Alice's score first and Bob's second.

Function Description

Complete the function compareTriplets in the editor below.

compareTriplets has the following parameter(s):

- int a[3]: Alice's challenge rating
- int b[3]: Bob's challenge rating

Return

- int[2]: Alice's score is in the first position, and Bob's score is in the second.

Input Format

The first line contains 3 space-separated integers, a[0], a[1], and a[2], the respective values in triplet a.

The second line contains 3 space-separated integers, b[0], b[1], and b[2], the respective values in triplet b.

Constraints

- $1 \le a[i] \le 100$
- $1 \le b[i] \le 100$

Sample Input 0

```
5 6 7
3 6 10
```

Sample Output 0

```
1 1
```

Explanation 0

In this example:

- $a = (a[0], a[1], a[2]) = (5, 6, 7)$
- $b = (b[0], b[1], b[2]) = (3, 6, 10)$

Now, let's compare each individual score:

- $a[0] > b[0]$, so Alice receives 1 point.
- $a[1] = b[1]$, so nobody receives a point.
- $a[2] < b[2]$, so Bob receives 1 point.

Alice's comparison score is 1, and Bob's comparison score is 1. Thus, we return the array $[1, 1]$.

Sample Input 1

```
17 28 30
99 16 8
```

Sample Output 1

```
2 1
```

Explanation 1

Comparing the 0th elements, $17 < 99$ so Bob receives a point.

Comparing the 1st and 2nd elements, $28 > 16$ and $30 > 8$ so Alice receives two points.

The return array is $[2, 1]$.

Solution:-

```
#include <bits/stdc+
+.h>using namespace
std;

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

vector<int> compareTriplets(vector<int> a, vector<int>

    b) {vector<int> result;

    int aliceScore =
    0;int bobScore =
    0;

    for(int i=0;i<a.size();i+
        +){if(a[i] > b[i])
            aliceScore ++;
        else if(b[i] >
        a[i])
            bobScore++;
        }
        result.push_back(aliceScore);
        result.push_back(bobScore);
    return result;
}


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

    string a_temp_temp;
    getline(cin,
    a_temp_temp);

    vector<string> a_temp =

    split(rtrim(a_temp_temp));vector<int> a(3);
```

```
    for (int i = 0; i < 3; i++) {
        int a_item =
        stoi(a_temp[i]);

        a[i] = a_item;
    }
    string b_temp_temp;
    getline(cin,
    b_temp_temp);

    vector<string> b_temp =

    split(rtrim(b_temp_temp));vector<int> b(3);

    for (int i = 0; i < 3; i++) {
        int b_item =
        stoi(b_temp[i]);

        b[i] = b_item;
    }

    vector<int> result = compareTriplets(a, b);

    for (size_t i = 0; i < result.size(); i+
        +) {fout << result[i];

        if (i != result.size() -
            1) {fout << " ";
        }
    }

    fout <<

    "\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;
}
```

Output:-

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

```
Success
```

Input (stdin) Download

```
5 6 7
3 6 10
```

Expected Output Download

```
1 1
```

Problem Statement 2.2 diagonal-difference

Given a square matrix, calculate the absolute difference between the sums of its diagonals.

For example, the square matrix arr is shown below:

```
1 2 3
4 5 6
9 8 9
```

The left-to-right diagonal = $1 + 5 + 9 = 15$. The right to left diagonal = $3 + 5 + 9 = 17$. Their absolute difference is $|15 - 17| = 2$.

Function description

Complete the $diagonalDifference$ function in the editor below.

diagonalDifference takes the following parameter:

- int arr[n][m]: an array of integers

Return

- int: the absolute diagonal difference

Input Format

The first line contains a single integer, n, the number of rows and columns in the square matrix arr.

Each of the next n lines describes a row, $arr[i]$, and consists of n space-separated integers $arr[i][j]$.

Constraints

- $-100 \leq arr[i][j] \leq 100$

Output Format

Return the absolute difference between the sums of the matrix's two diagonals as a single integer.

Sample Input

```
3
11 2 4
4 5 6
10 8 -12
```

Sample Output

```
15
```

Explanation

The primary diagonal is:

```
11
  5
    -12
```

Sum across the primary diagonal: 11 + 5 - 12 = 4

The secondary diagonal is:

```
    4
  5
10
```

Sum across the secondary diagonal: 4 + 5 + 10 = 19

Difference: |4 - 19| = 15

Note: |x| is the absolute value of x

Solution:

```
#include <cmath>
#include <cstdio>
#include <vector>
#include
<iostream>
#include
<algorithm>using
namespace std;


int main() {
    int N;
    cin >>
    N;
```

```
    int i, j;

    int sumdiag1 = 0;
int sumdiag2  =  0; for(i = 0; i < N; i++){
        for(j = 0; j< N; j++)
        {
           int no;
           cin >>
           no;if(i
           == j)
               sumdiag1 +=
           no;if(i+j == N-1)
              sumdiag2 += no;
        }
    }

    cout << abs(sumdiag1 -
    sumdiag2);return 0;
}
```

Output: