

Experiment Title : 2

Student Name: Rajiv Paul
Branch: CSE
Semester: 5
Subject Name: Competitive Coding-I

UID: 20BCS1812
Section/Group: 702 A
Date of Performance: 25/08/22
Subject Code: 20CSP-314

Problem Statement 2.1 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 $3 \rightarrow 2 \rightarrow 1 \rightarrow 0$. Hence, 3 moves.

For the case 2, we can either go $4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 0$ or $4 \rightarrow 2 \rightarrow 1 \rightarrow 0$. 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;

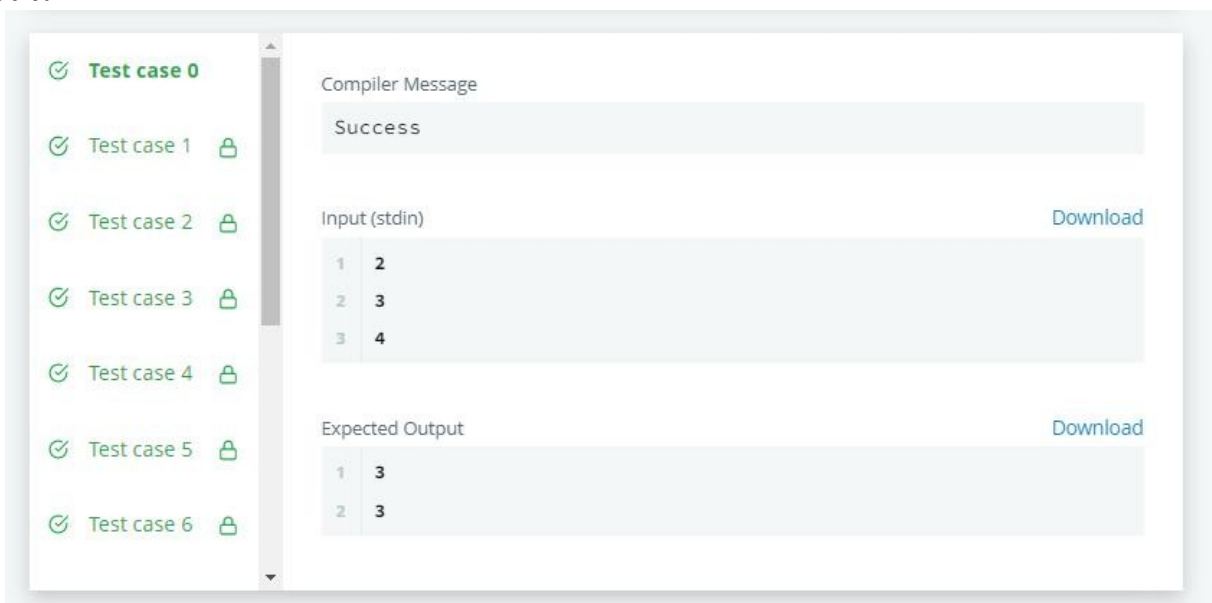
    for (int i=1; i < MAX; i+
        +) {int score = map[i]
        + 1; int limit;
        if (map[i+1] >
```

```
        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:



The screenshot displays a coding platform interface. On the left, a sidebar lists seven test cases, all marked as successful with green checkmarks. The main area is divided into three sections: 'Compiler Message' showing 'Success', 'Input (stdin)' with a table of three rows (2, 3, 4), and 'Expected Output' with a table of two rows (3, 3). Each section has a 'Download' link.

1	2
2	3
3	4

1	3
2	3

Problem Statement 2.2 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:

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

1	3
2	1 5
3	10 3
4	3 4

Expected Output

Download

1	1
---	---