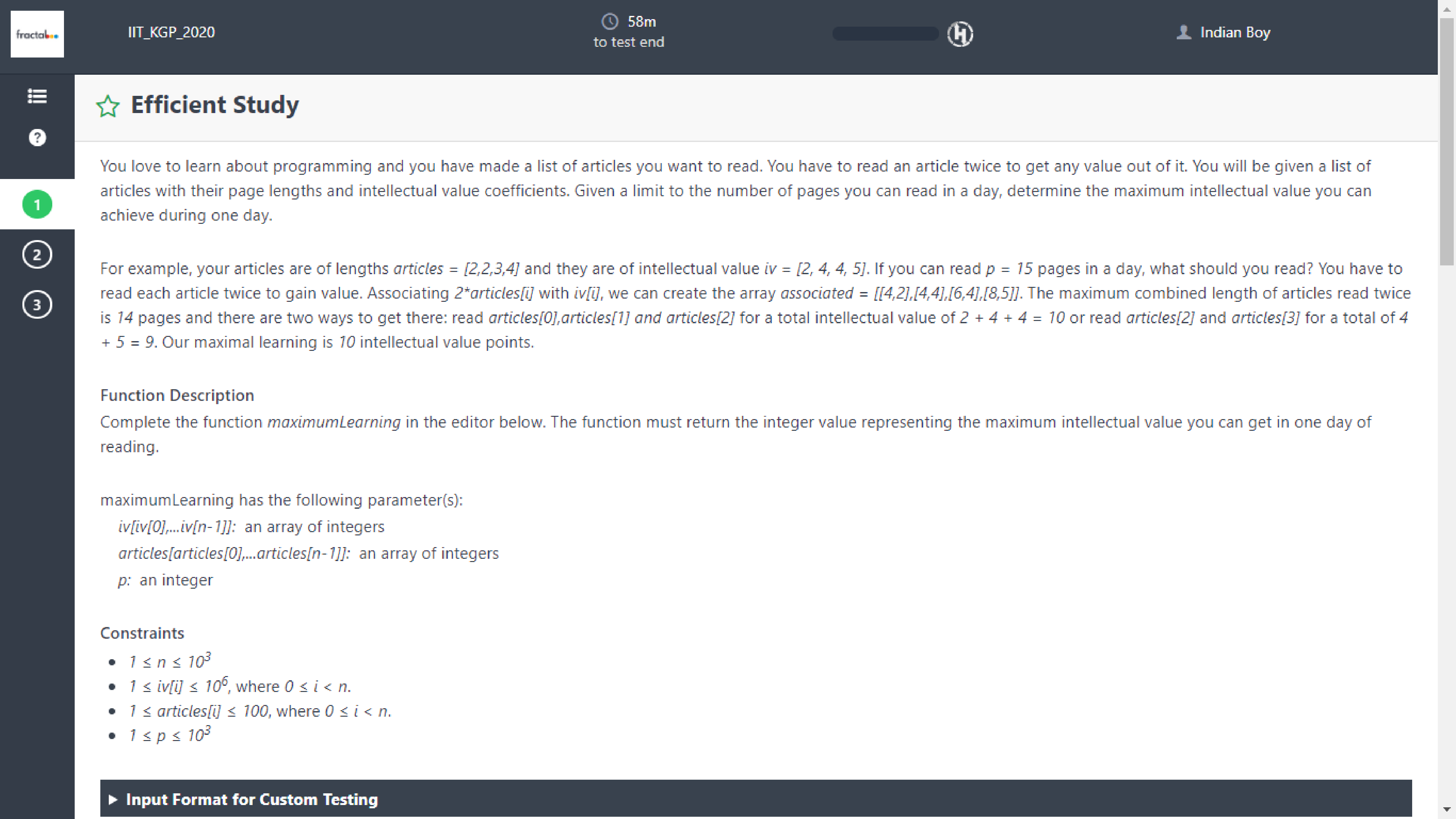


Efficient Study

You love to learn about programming and you have made a list of articles you want to read. You have to read an article twice to get any value out of it. You will be given a list of articles with their page lengths and intellectual value coefficients. Given a limit to the number of pages you can read in a day, determine the maximum intellectual value you can achieve during one day.

For example, your articles are of lengths *articles = [2,2,3,4]* and they are of intellectual value *iv = [2, 4, 4, 5]*. If you can read *p = 15* pages in a day, what should you read? You have to read each article twice to gain value. Associating *2*articles[i]* with *iv[i]*, we can create the array *associated = [[4,2],[4,4],[6,4],[8,5]]*. The maximum combined length of articles read twice is *14* pages and there are two ways to get there: read *articles[0],articles[1] and articles[2]* for a total intellectual value of *2 + 4 + 4 = 10* or read *articles[2]* and *articles[3]* for a total of *4 + 5 = 9*. Our maximal learning is *10* intellectual value points.

Function Description

Complete the function *maximumLearning* in the editor below. The function must return the integer value representing the maximum intellectual value you can get in one day of reading.

maximumLearning has the following parameter(s):

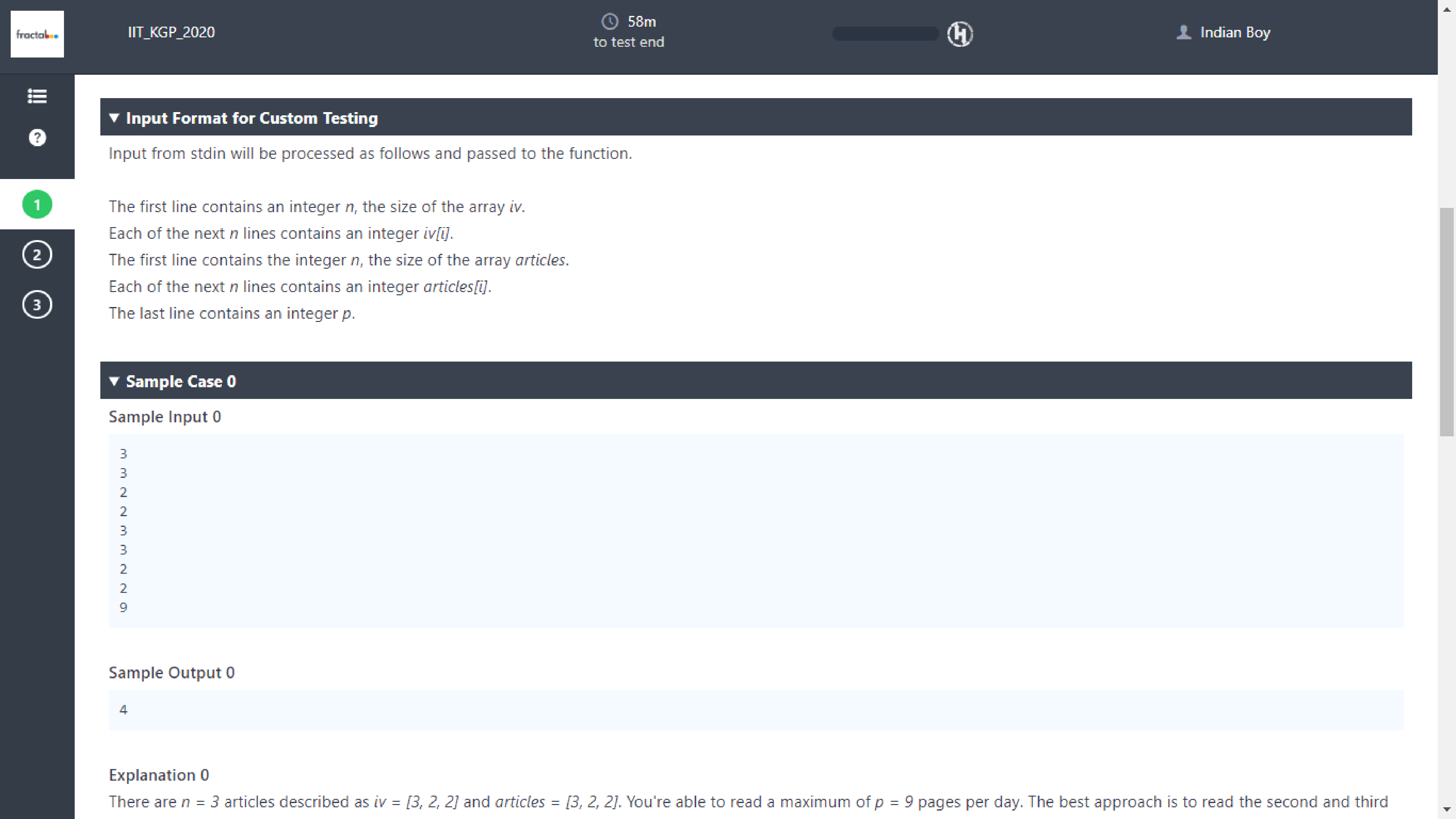
iv[iv[0],...iv[n-1]]: an array of integers

articles[articles[0],...articles[n-1]]: an array of integers

p: an integer

Constraints

- $1 \le n \le 10^3$
- $1 \le iv[i] \le 10^6$, where $0 \le i < n$.
- $1 \le articles[i] \le 100$, where $0 \le i < n$.
- $1 \le p \le 10^3$



▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the size of the array *iv*.
Each of the next *n* lines contains an integer *iv[i]*.
The first line contains the integer *n*, the size of the array *articles*.
Each of the next *n* lines contains an integer *articles[i]*.
The last line contains an integer *p*.

▼ Sample Case 0

Sample Input 0

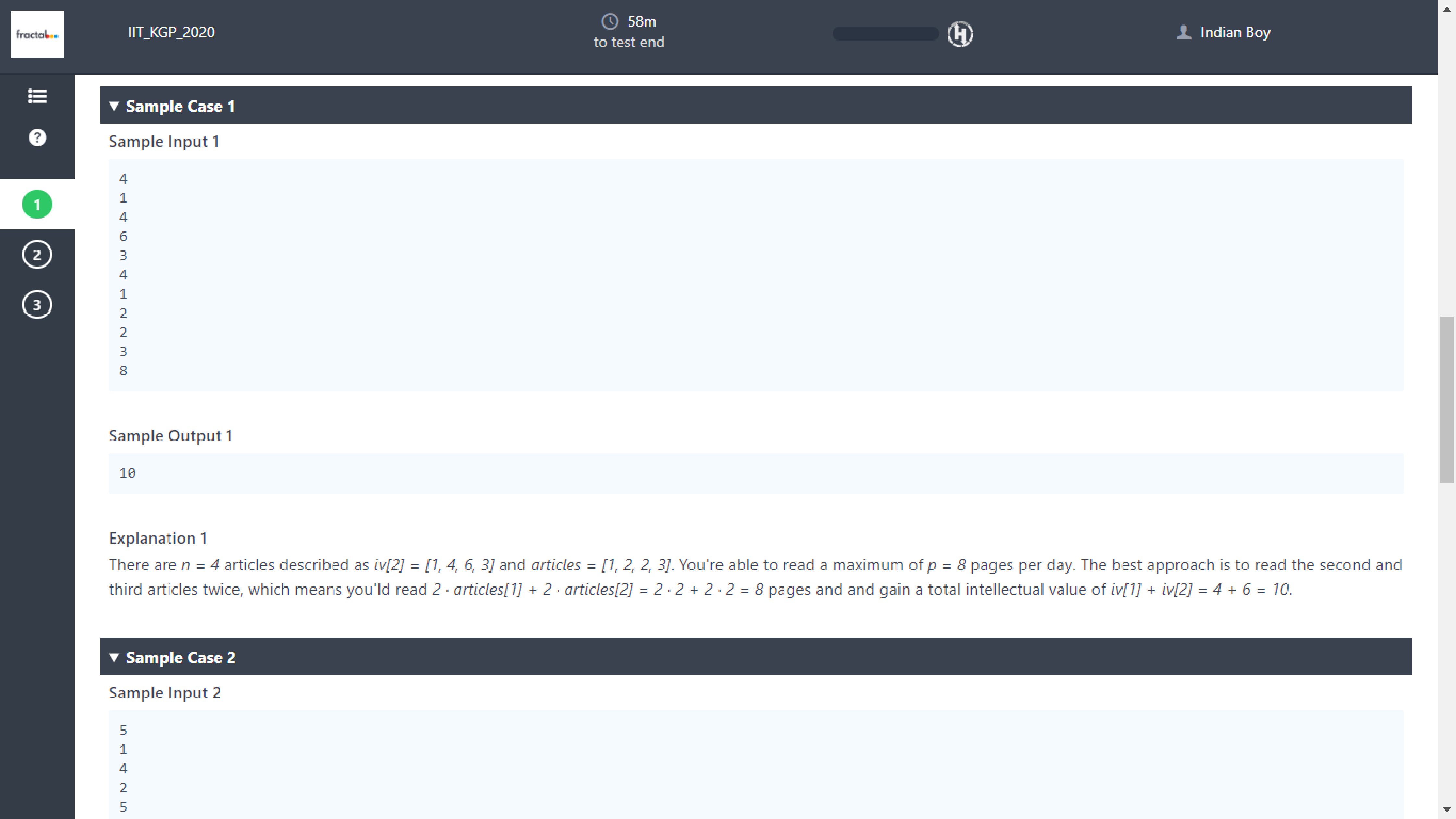
```
3
3
2
2
3
3
2
2
9
```

Sample Output 0

```
4
```

Explanation 0

There are *n* = *3* articles described as *iv* = *[3, 2, 2]* and *articles* = *[3, 2, 2]*. You're able to read a maximum of *p* = *9* pages per day. The best approach is to read the second and third



▼ Sample Case 1

Sample Input 1

```
4
1
4
6
3
4
1
2
2
3
8
```

Sample Output 1

```
10
```

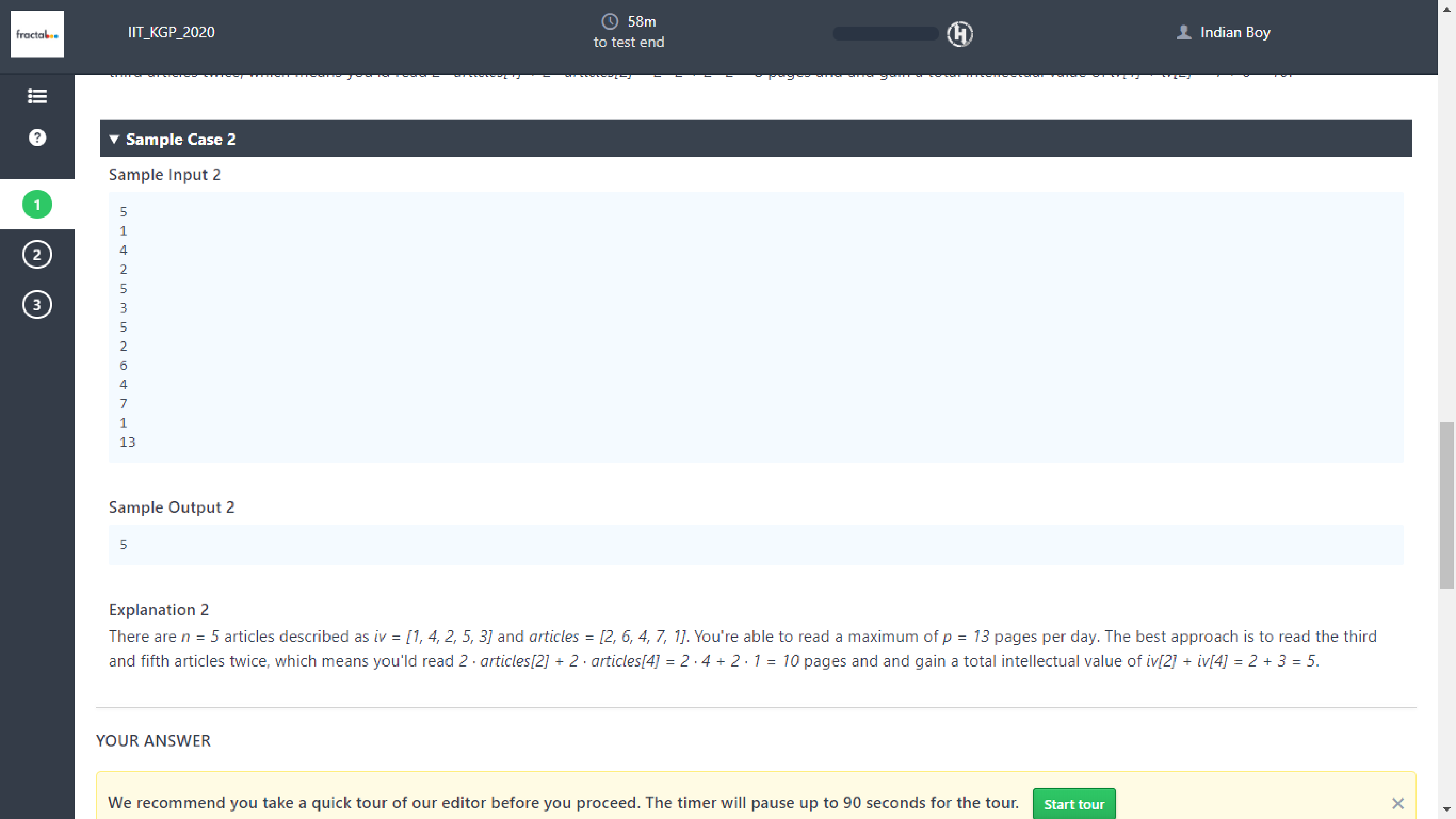
Explanation 1

There are $n = 4$ articles described as $iv[2] = [1, 4, 6, 3]$ and $articles = [1, 2, 2, 3]$. You're able to read a maximum of $p = 8$ pages per day. The best approach is to read the second and third articles twice, which means you'ld read $2 \cdot articles[1] + 2 \cdot articles[2] = 2 \cdot 2 + 2 \cdot 2 = 8$ pages and and gain a total intellectual value of $iv[1] + iv[2] = 4 + 6 = 10$.

▼ Sample Case 2

Sample Input 2

```
5
1
4
2
5
```



third articles twice, which means you'd read 2 · articles[1] + 2 · articles[2] = 2 · 2 + 2 · 2 = 8 pages and and gain a total intellectual value of iv[1] + iv[2] = 7 + 3 = 10.

▼ Sample Case 2

Sample Input 2

```
5
1
4
2
5
3
5
2
6
4
7
1
13
```

Sample Output 2

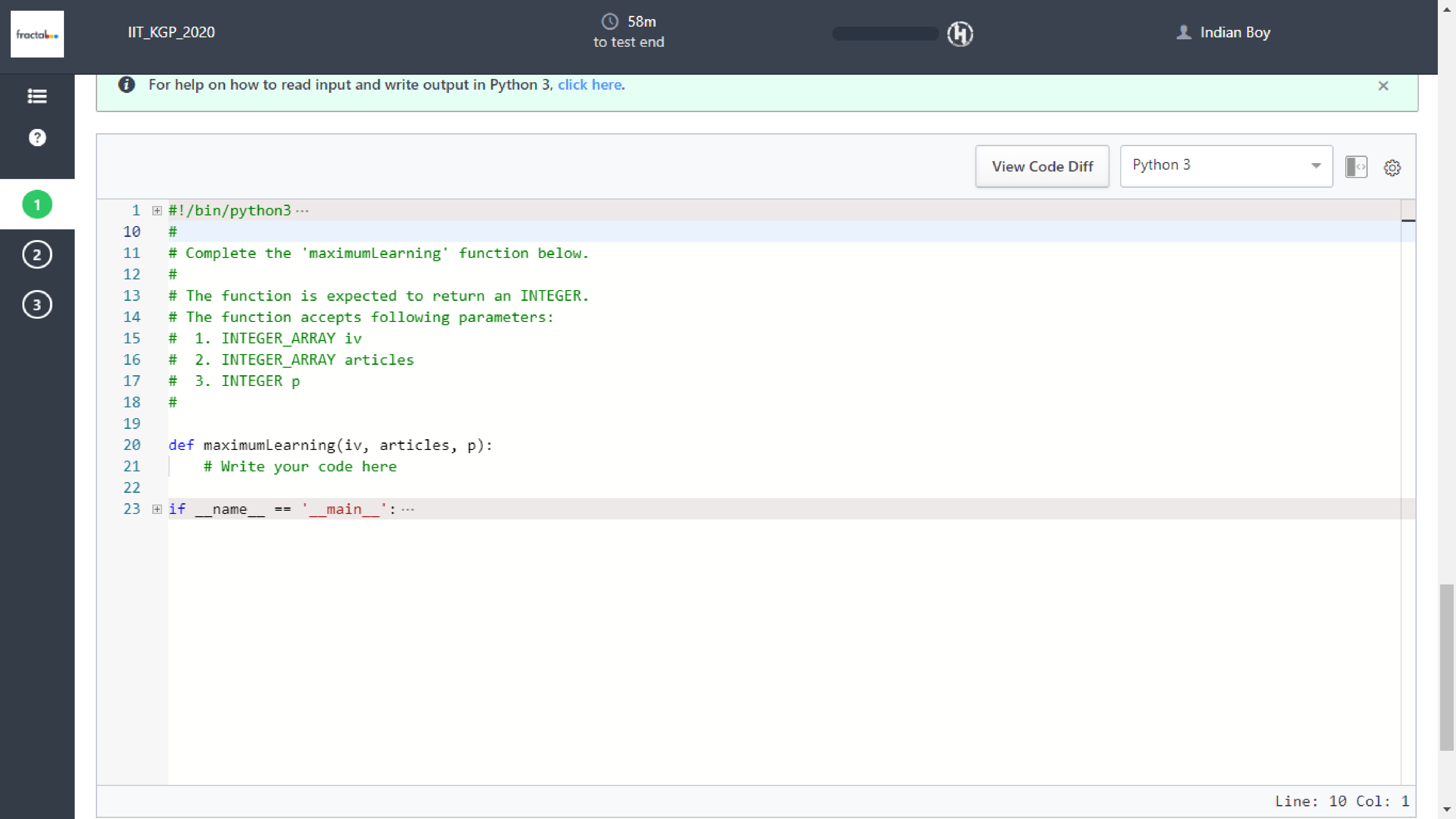
```
5
```

Explanation 2

There are *n = 5* articles described as *iv = [1, 4, 2, 5, 3]* and *articles = [2, 6, 4, 7, 1]*. You're able to read a maximum of *p = 13* pages per day. The best approach is to read the third and fifth articles twice, which means you'd read *2 · articles[2] + 2 · articles[4] = 2 · 4 + 2 · 1 = 10* pages and and gain a total intellectual value of *iv[2] + iv[4] = 2 + 3 = 5*.

YOUR ANSWER

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour. Start tour ✕

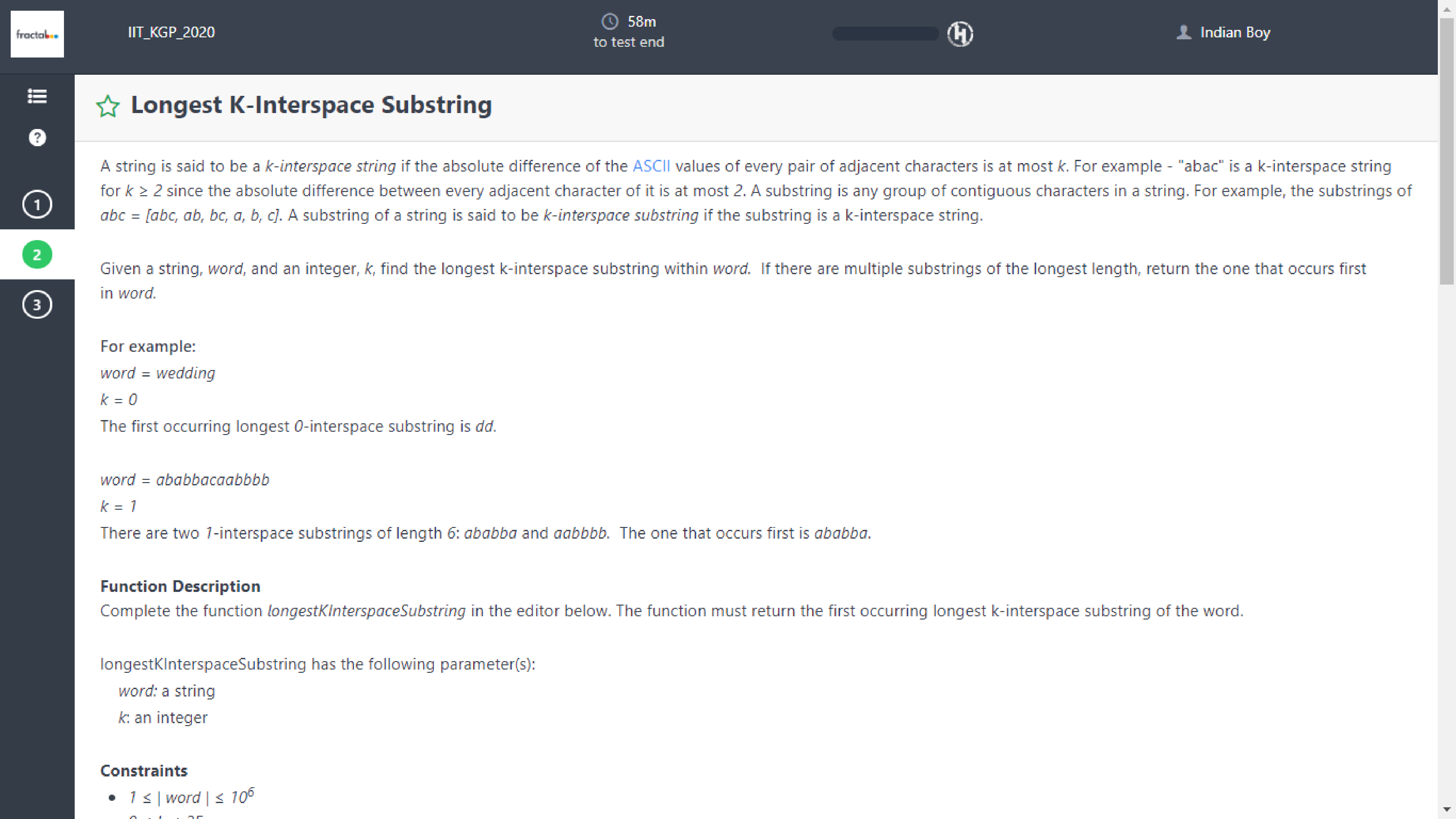


For help on how to read input and write output in Python 3, click here.

```
#!/bin/python3 ···
#
# Complete the 'maximumLearning' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. INTEGER_ARRAY iv
#  2. INTEGER_ARRAY articles
#  3. INTEGER p
#

def maximumLearning(iv, articles, p):
    # Write your code here

if __name__ == '__main__': ···
```



Longest K-Interspace Substring

A string is said to be a *k-interspace string* if the absolute difference of the ASCII values of every pair of adjacent characters is at most *k*. For example - "abac" is a k-interspace string for $k \geq 2$ since the absolute difference between every adjacent character of it is at most *2*. A substring is any group of contiguous characters in a string. For example, the substrings of *abc = [abc, ab, bc, a, b, c]*. A substring of a string is said to be *k-interspace substring* if the substring is a k-interspace string.

Given a string, *word*, and an integer, *k*, find the longest k-interspace substring within *word*. If there are multiple substrings of the longest length, return the one that occurs first in *word*.

For example:

word = wedding

k = 0

The first occurring longest *0*-interspace substring is *dd*.

word = ababbacaabbbb

k = 1

There are two *1*-interspace substrings of length *6*: *ababba* and *aabbbb*. The one that occurs first is *ababba*.

Function Description

Complete the function *longestKInterspaceSubstring* in the editor below. The function must return the first occurring longest k-interspace substring of the word.

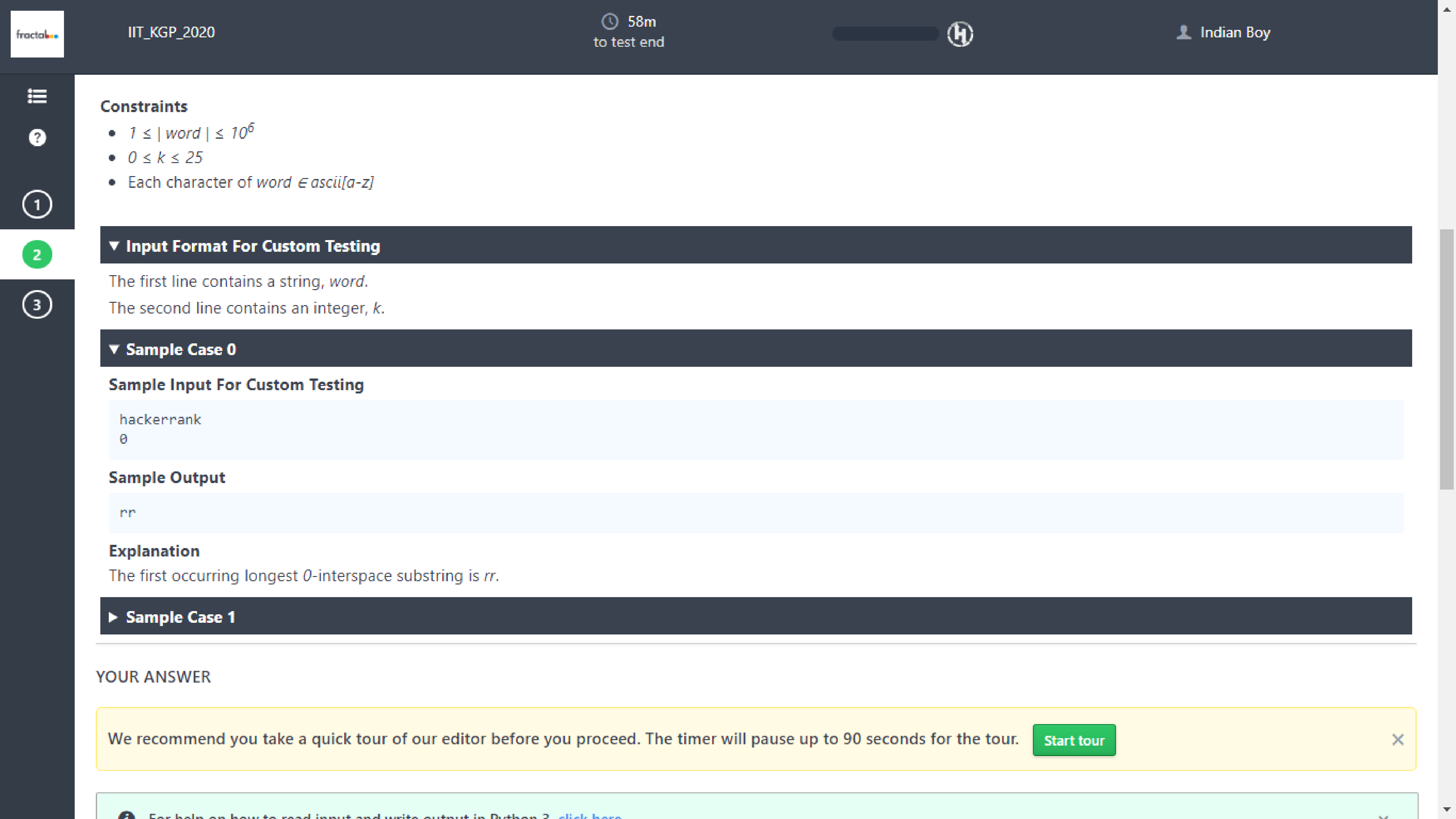
longestKInterspaceSubstring has the following parameter(s):

word: a string

k: an integer

Constraints

- $1 \leq |word| \leq 10^6$
- $0 \leq k \leq 25$



Constraints

- $1 \le |\ word\ | \le 10^6$
- $0 \le k \le 25$
- Each character of *word* $\in$ *ascii[a-z]*

▼ Input Format For Custom Testing

The first line contains a string, *word*.

The second line contains an integer, *k*.

▼ Sample Case 0

Sample Input For Custom Testing

```
hackerrank
0
```

Sample Output

```
rr
```

Explanation

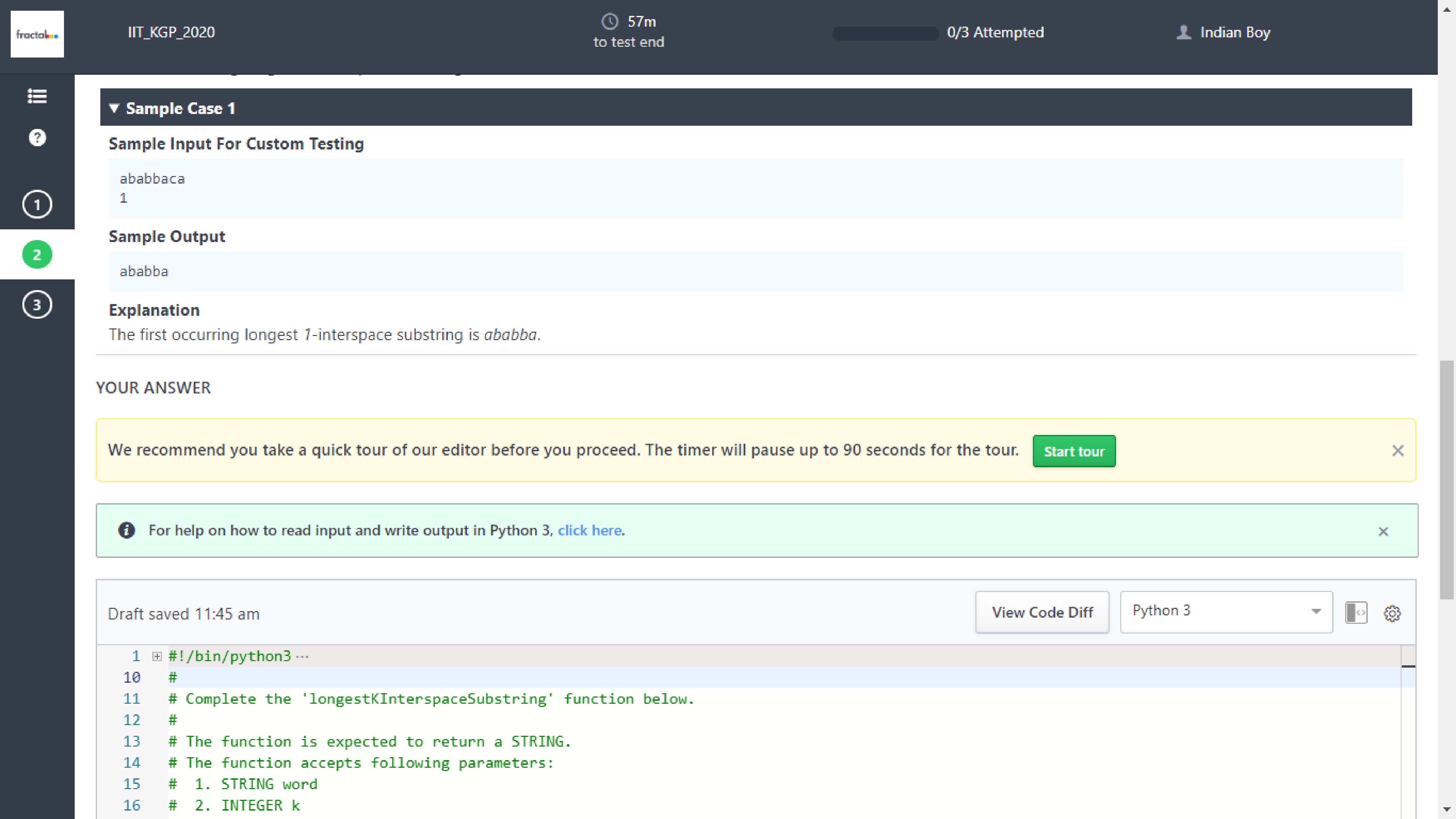
The first occurring longest *0*-interspace substring is *rr*.

▶ Sample Case 1

YOUR ANSWER

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour.

For help on how to read input and write output in Python 3, click here



▼ **Sample Case 1**

Sample Input For Custom Testing

```
ababbaca
1
```

Sample Output

```
ababba
```

Explanation

The first occurring longest *1*-interspace substring is *ababba*.

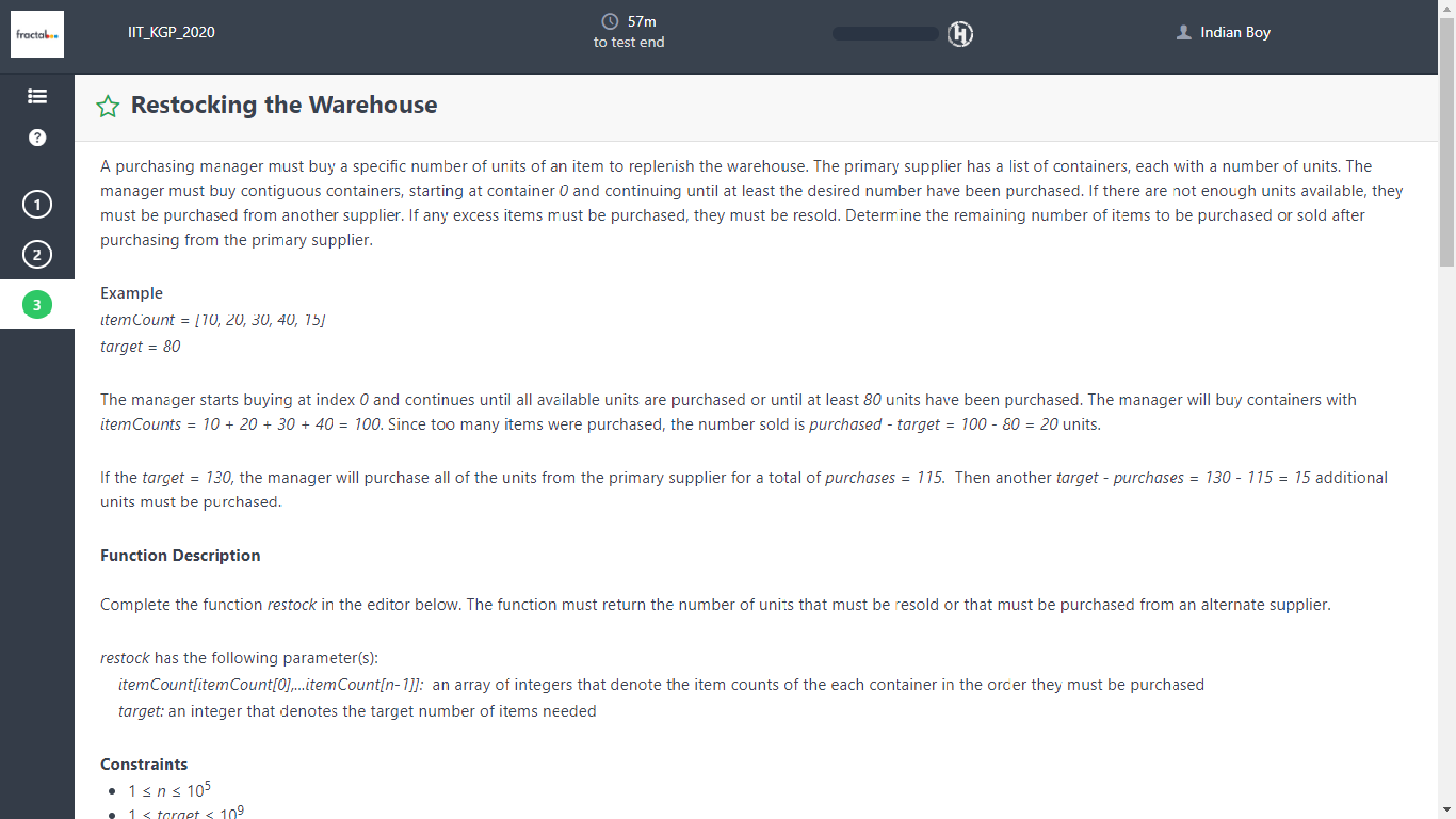
YOUR ANSWER

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour. Start tour ×

For help on how to read input and write output in Python 3, click here. ×

Draft saved 11:45 am

```
#!/bin/python3 ···
#
# Complete the 'longestKInterspaceSubstring' function below.
#
# The function is expected to return a STRING.
# The function accepts following parameters:
#  1. STRING word
#  2. INTEGER k
```



Restocking the Warehouse

A purchasing manager must buy a specific number of units of an item to replenish the warehouse. The primary supplier has a list of containers, each with a number of units. The manager must buy contiguous containers, starting at container *0* and continuing until at least the desired number have been purchased. If there are not enough units available, they must be purchased from another supplier. If any excess items must be purchased, they must be resold. Determine the remaining number of items to be purchased or sold after purchasing from the primary supplier.

Example

itemCount = [10, 20, 30, 40, 15]

target = 80

The manager starts buying at index *0* and continues until all available units are purchased or until at least *80* units have been purchased. The manager will buy containers with *itemCounts = 10 + 20 + 30 + 40 = 100*. Since too many items were purchased, the number sold is *purchased - target = 100 - 80 = 20* units.

If the *target = 130*, the manager will purchase all of the units from the primary supplier for a total of *purchases = 115*. Then another *target - purchases = 130 - 115 = 15* additional units must be purchased.

Function Description

Complete the function *restock* in the editor below. The function must return the number of units that must be resold or that must be purchased from an alternate supplier.

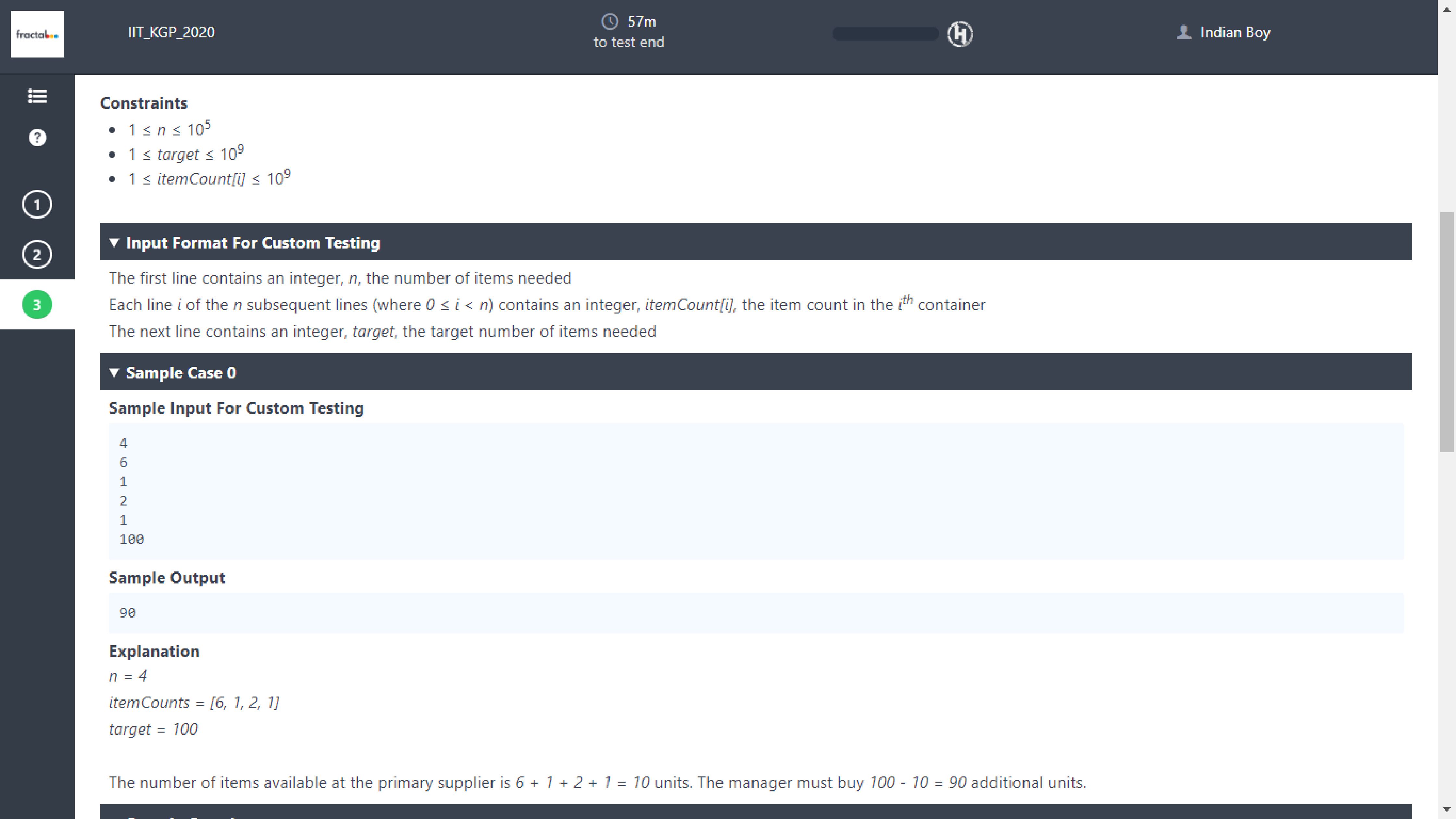
restock has the following parameter(s):

itemCount[itemCount[0],...itemCount[n-1]]: an array of integers that denote the item counts of the each container in the order they must be purchased

target: an integer that denotes the target number of items needed

Constraints

- $1 \le n \le 10^5$
- $1 \le target \le 10^9$



Constraints

- $1 \le n \le 10^5$
- $1 \le target \le 10^9$
- $1 \le itemCount[i] \le 10^9$

▼ Input Format For Custom Testing

The first line contains an integer, *n*, the number of items needed

Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains an integer, *itemCount[i]*, the item count in the i^{th} container

The next line contains an integer, *target*, the target number of items needed

▼ Sample Case 0

Sample Input For Custom Testing

```
4
6
1
2
1
100
```

Sample Output

```
90
```

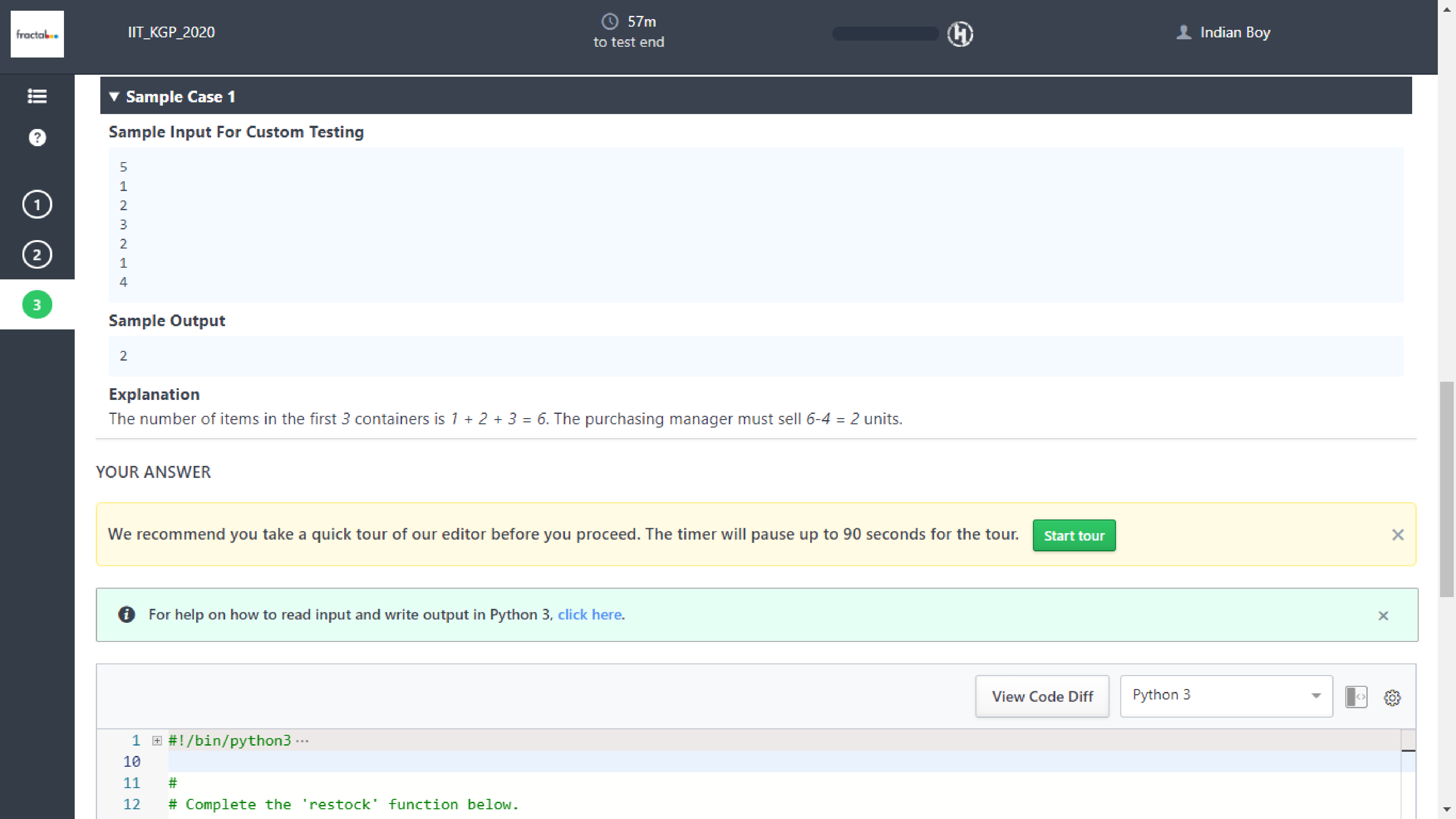
Explanation

n = 4

itemCounts = [6, 1, 2, 1]

target = 100

The number of items available at the primary supplier is *6 + 1 + 2 + 1 = 10* units. The manager must buy *100 - 10 = 90* additional units.



▼ **Sample Case 1**

Sample Input For Custom Testing

```
5
1
2
3
2
1
4
```

Sample Output

```
2
```

Explanation

The number of items in the first *3* containers is *1* + *2* + *3* = *6*. The purchasing manager must sell *6-4* = *2* units.

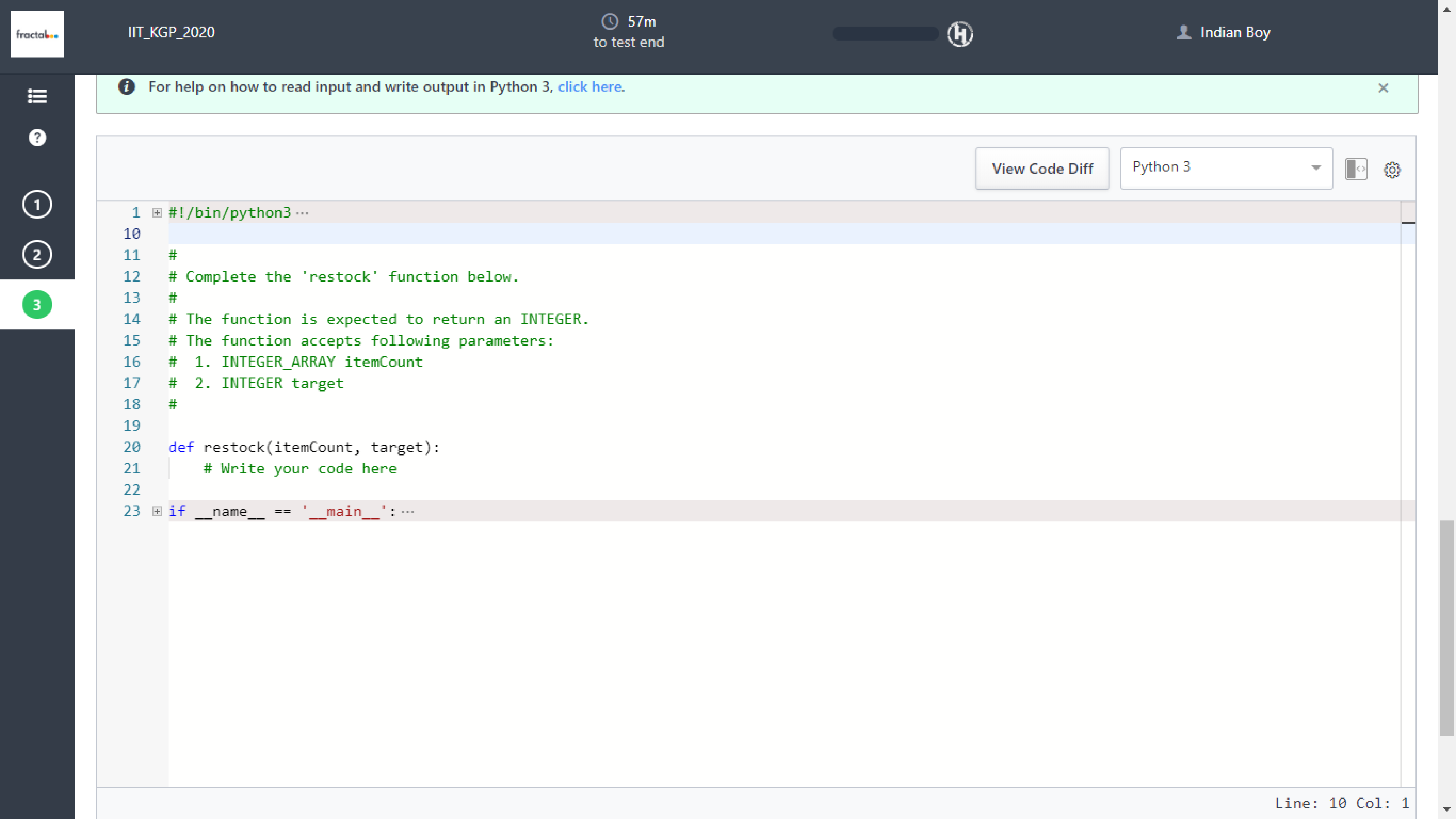
YOUR ANSWER

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour. Start tour ×

For help on how to read input and write output in Python 3, click here. ×

```
#!/bin/python3 ···

#
# Complete the 'restock' function below.
```



For help on how to read input and write output in Python 3, click here.

```
#!/bin/python3 ···

#
# Complete the 'restock' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. INTEGER_ARRAY itemCount
#  2. INTEGER target
#

def restock(itemCount, target):
    # Write your code here

if __name__ == '__main__': ···
```

Line: 10 Col: 1