You can view this report online at : https://www.hackerrank.com/x/tests/1742859/candidates/57751383/report

Full Name:	Instructor
Email:	aisha.batool@sse.habib.edu.pk
Test Name:	**CS101 - LW12 - Fall23**
Taken On:	5 Nov 2023 18:42:35 PKT
Time Taken:	5 min 7 sec/ 180 min
Student Roll Number:	02163
Section:	N/A
Invited by:	Aisha
Skills Score:	Problem Solving (Basic) 10/10
Tags Score:	Algorithms 10/10 Arrays 10/10 CS101 40/40 Easy 10/10 Lists 20/20 Python 3 20/20

100%

70/70

scored in **CS101 - LW12 - Fall23** in 5 min 7 sec on 5 Nov 2023 18:42:35 PKT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Theft Of Character > Coding	31 sec	10/ 10	✓
Q2	Container > Coding	36 sec	10/ 10	✓
Q3	Minimum Difference Sum > Coding	48 sec	10/ 10	✓
Q4	Sum 67 > Coding	26 sec	10/ 10	✓
Q5	Wanna (reverse a) piece of me? > Coding	34 sec	10/ 10	✓
Q6	Noisy neighbors > Coding	42 sec	10/ 10	✓
Q7	Hindu shuffle > Coding	1 min 4 sec	10/ 10	✓
Q8	Difficulty Meter > Multiple Choice	13 sec	0/ 0	✓

QUESTION 1 ✓

Theft Of Character > Coding

Correct Answer

Score 10

QUESTION DESCRIPTION

Problem

Someone is stealing characters from your strings. They only steal one character at a time and leave a `_` in place of the stolen character. With the missing character, it is difficult for you to recall the original string from the remains of the theft. You decide to precompute all possible thefts so that you can identify the missing letter when the theft happens.

Write a function named `blanks` that removes each letter from its parameter `s` one by one, concatenates it with remaining string in the same format as given below in the example, and then concatenates all these strings generated and returns them. Printing will happen at function call.

Sample

```
>>> s = blanks('')
>>> print(s)
''

>>> s = blanks('12 Gf!')
>>> print(s)
'1 : _2 Gf!
2 : 1_ Gf!
  : 12_Gf!
G : 12 _f!
f : 12 G_!
! : 12 Gf_'
```

Constraints

- `isinstance(s, str)` is `True`
- `s` does not contain repeated characters.

INTERVIEWER GUIDELINES

Solution

```
def blanks(s):
    str = ""
    for i in range(len(s)):
        str += s[i] + ' : ' + s[:i]+'_'+s[i+1:] + "\n"

    return str
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1 def blanks(s):
2     str = ""
3     for i in range(len(s)):
4         str += s[i] + ' : ' + s[:i]+'_'+s[i+1:] + "\n"
5
6     return str
7
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	1	0.0491 sec	9.23 KB
Testcase 1	Easy	Sample case	Success	1.8	0.0468 sec	9.41 KB
Testcase 2	Easy	Sample case	Success	1.8	0.0658 sec	9.58 KB

Testcase 4	Easy	Hidden case	Success	1.8	0.0781 sec	9.3 KB
Testcase 5	Easy	Hidden case	Success	1.8	0.0485 sec	9.6 KB
Testcase 6	Easy	Hidden case	Success	1.8	0.1758 sec	9.26 KB

No Comments

QUESTION 2

Correct Answer

Score 10

Container › Coding Lists CS101

QUESTION DESCRIPTION

Problem

Write a function that, given two lists, determines whether the second list is a sublist of the first.

Sample

```
>>> is_sublist([0,1,2,3,4], [1,2,3])
True
>>> is_sublist([0,1,2,3,4], [1,2,4])
False
>>> is_sublist([0,1,2,3,4], [0,1,2,3,4,5])
False
```

Input Format

The input contains the two lists on separate lines. Each line contains a space separated list.

INTERVIEWER GUIDELINES

Solution

```
def is_sublist(lst, sub):
    """Is sub a sublist of lst?

    Args:
    - lst ([]): the list to check
    - sub ([]): do the elements of sub appear in the same order in lst?

    Observations:
    - The elements of sub need to be appear in the same order in lst.
    - Answer the question by checking all slices of lst of the size of
sub.
    - If a slice mathces sub, the answer is yes.

    Returns:
    bool: do the elements of sub appear in the same order in lst?
    """
    # Match all slices of lst of the right size with sub.
    n = len(lst)
    k = len(sub)
    for i in range(n-k):
        if lst[i:i+k] == sub:  # match found!
            return True
    # Failed to find a match.
    return False


# Read in strings and break them into lists.
lst = input().strip().split()
sub = input().strip().split()
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def is_sublist(lst, sub):
    """Is sub a sublist of lst?

    Args:
    - lst ([]): the list to check
    - sub ([]): do the elements of sub appear in the same order in lst?

    Observations:
    - The elements of sub need to be appear in the same order in lst.
    - Answer the question by checking all slices of lst of the size of sub.
    - If a slice mathces sub, the answer is yes.

    Returns:
    bool: do the elements of sub appear in the same order in lst?
    """
    # Match all slices of lst of the right size with sub.
    n = len(lst)
    k = len(sub)
    for i in range(n-k):
        if lst[i:i+k] == sub:  # match found!
            return True
    # Failed to find a match.
    return False


# Read in strings and break them into lists.
lst = input().strip().split()
sub = input().strip().split()

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	2.5	0.0528 sec	9.34 KB
Testcase 1	Easy	Sample case	Success	2.5	0.07 sec	9.48 KB
Testcase 2	Easy	Hidden case	Success	2.5	0.1312 sec	9.26 KB
Testcase 3	Easy	Sample case	Success	2.5	0.0867 sec	9.48 KB

No Comments

QUESTION 3

Correct Answer

Score 10

Minimum Difference Sum › Coding

Easy | Arrays | Algorithms

QUESTION DESCRIPTION

Given an array of *n* integers, rearrange them so that the sum of the absolute differences of all adjacent elements is minimized. Then, compute the sum of those absolute differences.

Example

n = 5

arr = [1, 3, 3, 2, 4]

If the list is rearranged as *arr' = [1, 2, 3, 3, 4]*, the absolute differences are *|1 - 2| = 1, |2 - 3| = 1, |3 - 3| = 0, |3 - 4| = 1*. The sum of those differences is *1 + 1 + 0 + 1 = 3*.

Function Description

Complete the function *minDiff* in the editor below.

minDiff has the following parameter:

arr: an integer array

Returns:

int: the sum of the absolute differences of adjacent elements

Constraints

- $2 \le n \le 10^5$
- $0 \le arr[i] \le 10^9$, *where* $0 \le i < n$

▼ Input Format For Custom Testing

The first line of input contains an integer, *n,* the size of *arr*.

Each of the following *n* lines contains an integer that describes *arr[i]* (where $0 \le i < n$) .

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN    Function
-----    --------
5    →   arr[] size n = 5
5    →   arr[] = [5, 1, 3, 7, 3]
1
3
7
3
```

Sample Output

```
6
```

Explanation

n = 5

arr = [5, 1, 3, 7, 3]

If *arr* is rearranged as *arr' = [1, 3, 3, 5, 7]*, the differences are minimized.

The final answer is *|1 - 3| + |3 - 3| + |3 - 5| + |5 - 7| = 6*.

▼ Sample Case 1

Sample Input For Custom Testing

```
STDIN    Function
-----    --------
2    →   arr[] size n = 2
3    →   arr[] = [3, 2]
2
```

Sample Output

```
1
```

Explanation

n = 2

arr = [3, 2]

There is no need to rearrange because there are only two elements. The final answer is *|3 - 2| = 1*.

INTERVIEWER GUIDELINES

```
def minDiff(arr):
    sum = 0
    arr = sorted(arr)
    for i in range(len(arr)-1):
        sum +=abs(arr[i] - arr[i+1])
    return sum
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1
2  #
3  # Complete the 'minDiff' function below.
4  #
5  # The function is expected to return an INTEGER.
6  # The function accepts INTEGER_ARRAY arr as parameter.
7  #
8
9  def minDiff(arr):
10     sum = 0
11     arr = sorted(arr)
12     for i in range(len(arr)-1):
13         sum +=abs(arr[i] - arr[i+1])
14     return sum
15
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	Success	0.7	0.0954 sec	10.7 KB
TestCase 1	Easy	Sample case	Success	0.7	0.059 sec	10.5 KB
TestCase 2	Easy	Sample case	Success	0.7	0.0728 sec	10.7 KB
TestCase 3	Easy	Sample case	Success	0.7	0.0906 sec	10.9 KB
TestCase 4	Easy	Sample case	Success	0.7	0.0688 sec	10.8 KB
TestCase 5	Easy	Hidden case	Success	0.7	0.0663 sec	10.9 KB
TestCase 6	Easy	Hidden case	Success	0.7	0.23 sec	15.7 KB
TestCase 7	Easy	Hidden case	Success	0.7	0.1774 sec	15.5 KB
TestCase 8	Easy	Hidden case	Success	0.7	0.1985 sec	15.7 KB
TestCase 9	Easy	Hidden case	Success	0.7	0.1759 sec	15.4 KB
TestCase 10	Easy	Hidden case	Success	0.7	0.2246 sec	15.6 KB
TestCase 11	Easy	Hidden case	Success	0.7	0.1895 sec	15.5 KB
TestCase 12	Easy	Hidden case	Success	0.8	0.2025 sec	15.8 KB
TestCase 13	Easy	Hidden case	Success	0.8	0.2129 sec	15.7 KB

No Comments

QUESTION 4

Correct Answer

Score 10

Sum 67 › Coding CS101 Lists

QUESTION DESCRIPTION

Problem

Write a function named `sum67` that returns the sum of the numbers in its parameter named `lst` except that it ignores sections of numbers starting with a 6 and extending to the next 7 (every 6 will be followed by at least one 7). 0 is returned for no numbers.

Sample

```
>>> sum67([1, 2, 2])
5
>>> sum67([1, 2, 2, 6, 99, 99, 7])
5
>>> sum67([1, 1, 6, 7, 2])
4
```

Input Format

The input contains space separated numbers. These numbers in the given order are to be passed on as a list to `sum67`.

Constraints

The input contains integer values only.

INTERVIEWER GUIDELINES

```
def sum67(lst):
    """Returns the sum of lst ignoring subsequences from between 6 and 7.

    Args:
    - lst ([int]): the contained numbers are to be summed except for
      subsequences beginning with 6 and ending in 7.

    Observations:
    - The opening 6 and closing 7 are part of the seuqence to be ignored.

    Returns:
    int: the sum of numbers in lst except for the subsequences to be
ignored.
    """
    total = 0
    ignore = False
    for n in lst:
        if n == 6:
            ignore = True
        if not ignore:
            total += n
        if n == 7:
            ignore = False
    return total


# Input the list as str, split on whitespace, and convert each str to
int.
lst = input().strip().split()
lst = [int(n) for n in lst]
```

CANDIDATE ANSWER

Language used: **Python 3**

```python
def sum67(lst):
    """Returns the sum of lst ignoring subsequences from between 6 and 7.

    Args:
    - lst ([int]): the contained numbers are to be summed except for
      subsequences beginning with 6 and ending in 7.

    Observations:
    - The opening 6 and closing 7 are part of the seuqence to be ignored.

    Returns:
    int: the sum of numbers in lst except for the subsequences to be ignored.
    """
    total = 0
    ignore = False
    for n in lst:
        if n == 6:
            ignore = True
        if not ignore:
            total += n
        if n == 7:
            ignore = False
    return total


# Input the list as str, split on whitespace, and convert each str to int.
lst = input().strip().split()
lst = [int(n) for n in lst]

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	1.5	0.0886 sec	9.51 KB
Testcase 1	Easy	Hidden case	Success	1.5	0.075 sec	9.7 KB
Testcase 2	Easy	Hidden case	Success	2	0.1212 sec	9.36 KB
Testcase 3	Easy	Sample case	Success	1.5	0.0572 sec	9.42 KB
Testcase 4	Easy	Sample case	Success	1.5	0.054 sec	9.33 KB
Testcase 5	Easy	Hidden case	Success	2	0.0523 sec	9.54 KB

No Comments

QUESTION 5

Correct Answer

Score 10

Wanna (reverse a) piece of me? › Coding

QUESTION DESCRIPTION

Challenge

In Python, lists have a built-in method called `reverse` that reverses **all** elements in a list **in place**.

Write a function called `reverse_slice` that takes a list `t` and two indexes, `start` and `stop`, as its arguments, and reverses all elements in the list slice `t[start:stop]` **in place**. The function **does not return anything useful** to stress the fact that it modifies the list. If the list slice is empty, list `t` is left unmodified.

Note

List indexes may be zero, positive, or negative, or may even indicate a slice that is empty. Your solution must handle all such situations. The order of the remaining elements in the list must be preserved.

Sample interaction

```
>>> t = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> reverse_slice(t, 3, 7)
>>> t
[0, 1, 2, 6, 5, 4, 3, 7, 8, 9]
>>> t = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> reverse_slice(t, 2, -4)
>>> t
[0, 1, 5, 4, 3, 2, 6, 7, 8, 9]
>>> t = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> reverse_slice(t, 0, 10)
>>> t
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
>>> t = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> reverse_slice(t, 0, -1)
>>> t
[8, 7, 6, 5, 4, 3, 2, 1, 0, 9]
>>> t = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> reverse_slice(t, 20, 30)
>>> t
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> t = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> reverse_slice(t, -20, -10)
>>> t
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Input/Output

Your function will be provided the list and the index arguments, `t`, `start`, and `stop`, respectively. The returned value from your function, as well as its side-effect on the list `t`, will be checked by HackerRank.

Constraints

None

INTERVIEWER GUIDELINES

Solution

```
# Using the list.reverse method:
def reverse_slice(t, start, stop):
    slice = t[start:stop]
    slice.reverse()
    t[start:stop] = slice

# Using the reversed builtin "function":
def reverse_slice(t, start, stop):
    t[start:stop] = reversed(t[start:stop])
```

CANDIDATE ANSWER

Language used: **Python 3**

```
# Using the list.reverse method:
```

```
# using the list.reverse method.
def reverse_slice(t, start, stop):
    slice = t[start:stop]
    slice.reverse()
    t[start:stop] = slice


```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	0.66	0.0538 sec	9.43 KB
Testcase 1	Easy	Sample case	Success	0.66	0.069 sec	9.44 KB
Testcase 2	Easy	Sample case	Success	0.66	0.0636 sec	9.52 KB
Testcase 3	Easy	Sample case	Success	0.66	0.0436 sec	9.63 KB
Testcase 4	Easy	Sample case	Success	0.66	0.0377 sec	9.55 KB
Testcase 5	Easy	Sample case	Success	0.67	0.0404 sec	9.67 KB
Testcase 6	Easy	Hidden case	Success	0.67	0.0553 sec	9.49 KB
Testcase 7	Easy	Hidden case	Success	0.67	0.0895 sec	9.21 KB
Testcase 8	Easy	Hidden case	Success	0.67	0.0685 sec	9.29 KB
Testcase 9	Easy	Hidden case	Success	0.67	0.0995 sec	9.34 KB
Testcase 10	Easy	Hidden case	Success	0.67	0.0662 sec	9.45 KB
Testcase 11	Easy	Hidden case	Success	0.67	0.0558 sec	9.55 KB
Testcase 12	Easy	Sample case	Success	0.67	0.0454 sec	9.23 KB
Testcase 13	Easy	Sample case	Success	0.67	0.0652 sec	9.51 KB
Testcase 14	Easy	Sample case	Success	0.67	0.1302 sec	9.65 KB

No Comments

QUESTION 6

Correct Answer

Score 10

Noisy neighbors › Coding

CS101 | Python 3

QUESTION DESCRIPTION

Background

Netiquettes, the unofficial rules governing online interactions, dictate that one must not SHOUT, that is, write in all caps. Let us declare as noisy all such strings that have letters of the alphabet which are in all caps.

Task

Given a list of strings, find all the neighbors of noisy strings within that list.

Function Description

To implement the given task, write the following function:

1. ***noisy_neighbors*** that takes one argument: ***t***. The function should **return** a list that contains neighbors of noisy strings in ***t***. Note that each neighbor may only appear once in the same order as in ***t***.

Constraints

- Input and output will be handled by HackerRank--you should not read input or display values yourself.
- The list *t* will only contain zero or more strings.
- Each string in list *t* will contain at least one letter of the alphabet.

▼ Input Format For Custom Testing

The first and only line contains *t*, a list of strings.

▼ Sample Case 0

Sample Input For Custom Testing

```
['how', 'much', 'wood', 'would', 'a', 'WOODCHUCK', 'CHUCK', 'if', 'a',
'woodchuck', 'could', 'CHUCK', 'WOOD']
```

Sample Output

```
['a', 'WOODCHUCK', 'CHUCK', 'if', 'could', 'CHUCK', 'WOOD']
```

Explanation

The neighbors of 'WOODCHUCK' in position 5 are 'a' and 'CHUCK' in positions 4 and 6, respectively.
The neighbors of 'CHUCK' in position 6 are 'WOODCHUCK' and 'if' in positions 5 and 7, respectively.
The neighbors of 'CHUCK' in position 11 are 'could' and 'WOOD' in positions 10 and 12, respectively.
The only neighbor of 'WOOD' in position 12 is 'CHUCK' in positions 11.
These seven neighbors are arranged in the same order as the original list.

▼ Sample Case 1

Sample Input For Custom Testing

```
['HOW', 'much', 'WOOD', 'would', 'a', 'woodchuck', 'chuck', 'if', 'a',
'woodchuck', 'could', 'chuck', 'wood']
```

Sample Output

```
['much', 'would']
```

Explanation

The only neighbor of 'HOW' in position 0 is 'much' in positions 1.
The neighbors of 'WOOD' in position 2 are 'much' and 'would' in positions 1 and 3, respectively.
Since 'much' in position 1 is a common neighbor to the two words, it is only included once.
The two neighbors are arranged in the same order as the original list.

INTERVIEWER GUIDELINES

```
def noisy_neighbors(t):
    result = []
    for pos in range(len(t)):
        if (pos > 0 and t[pos - 1].isupper() or
            pos < len(t) - 1 and t[pos + 1].isupper()):
            result.append(t[pos])
    return result
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1 def noisy_neighbors(t):
2     result = []
3     for pos in range(len(t)):
4         if (pos > 0 and t[pos - 1].isupper() or
```

```
            pos < len(t) - 1 and t[pos + 1].isupper()):
            result.append(t[pos])
    return result

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	Success	1	0.0442 sec	10.1 KB
TestCase 1	Easy	Sample case	Success	1	0.1563 sec	10.1 KB
TestCase 2	Easy	Sample case	Success	1	0.1045 sec	10.1 KB
TestCase 3	Easy	Sample case	Success	1	0.0704 sec	10.2 KB
TestCase 4	Easy	Sample case	Success	1	0.0935 sec	10.2 KB
TestCase 5	Easy	Hidden case	Success	1	0.0941 sec	10.1 KB
TestCase 6	Easy	Hidden case	Success	1	0.054 sec	10.2 KB
TestCase 7	Easy	Hidden case	Success	1	0.1046 sec	10.1 KB
TestCase 8	Easy	Hidden case	Success	1	0.0514 sec	10.4 KB
TestCase 9	Easy	Hidden case	Success	1	0.0815 sec	10.1 KB

No Comments

QUESTION 7

Correct Answer

Score 10

Hindu shuffle › Coding

CS101 Python 3

QUESTION DESCRIPTION

Background

A deck of cards is hindu shuffled by cutting the deck from the top, and moving the cut to the bottom of the deck. A perfect hindu shuffle always cuts the same number of cards from the top of the deck.

In Python, a deck may be represented as a list of items. For example, if the list is [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], and each cut has a 3 cards, it gets cut into [1, 2, 3], then [4, 5, 6], then [7, 8, 9], and finally [10]. The final cut may have fewer cards than the cut requires. The first cut is moved to the bottom of the deck, second cut just above the first, and so on. This generates the list [10, 7, 8, 9, 4, 5, 6, 1, 2, 3].

Task

Write a function that accepts a list of items, and hindu shuffles the items into a new permutation.

Function Description

To implement the given task, write the following function:

1. ***hindu_shuffle*** that takes two arguments: ***deck*** of type list, **cut** of type *int*. The function should modify the list ***deck*** so that it is hindu shuffled. No useful value needs to be returned.

Notes and Constraints

- Input and output will be handled by HackerRank--you should not read input or display values yourself.
- HackerRank will display the original list. Returned value will not be displayed.
- List may have 0 or more items.
- Your function must modify the original list ***t***.
- Your function must not use any global variables, or import any modules.

▼ Input Format For Custom Testing

The first and only line contains *t*, a list.

▼ Sample Case 0

Sample Input For Custom Testing

```
['aardvark', 'buffalo', 'crocodile', 'dingo', 'elephant', 'fox',
'gorilla', 'hippopotamus']
4
```

Sample Output

```
['elephant', 'fox', 'gorilla', 'hippopotamus', 'aardvark', 'buffalo',
'crocodile', 'dingo']
```

Explanation

The list is cut 4 cards at a time, ['aardvark', 'buffalo', 'crocodile', 'dingo'] and ['elephant', 'fox', 'gorilla', 'hippopotamus']. First cut goes to the bottom of the deck, and second cut goes above the first cut.

Hint: If you need to store the original list in another variable, you can use list's built-in method **copy()** to shallow-copy the list to another variable.

```
>> lis = ['A','P','S']
>> new_list = lis.copy()
>> new_list
['A', 'P', 'S']

>> lis.pop()

>> new_list
['A', 'P', 'S']
>> lis
['A', 'P']
```

INTERVIEWER GUIDELINES

```
def hindu_shuffle(deck, cut):
    dup = deck.copy()
    for i in range(0, len(deck), cut):
        if i > 0:
            deck[-i-cut:-i] = dup[i:i+cut]
        else:
            deck[-cut:] = dup[:cut]
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def hindu_shuffle(deck, cut):
    dup = deck.copy()
    for i in range(0, len(deck), cut):
        if i > 0:
            deck[-i-cut:-i] = dup[i:i+cut]
        else:
            deck[-cut:] = dup[:cut]

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	Success	1	0.1838 sec	11.6 KB
TestCase 1	Easy	Sample case	Success	1	0.0655 sec	11.7 KB
TestCase 2	Easy	Sample case	Success	1	0.1253 sec	11.7 KB
TestCase 3	Easy	Sample case	Success	1	0.0629 sec	11.8 KB
TestCase 4	Easy	Sample case	Success	1	0.1028 sec	11.7 KB
TestCase 5	Easy	Hidden case	Success	1	0.0785 sec	11.6 KB
TestCase 6	Easy	Hidden case	Success	1	0.1053 sec	11.9 KB
TestCase 7	Easy	Hidden case	Success	1	0.068 sec	11.7 KB
TestCase 8	Easy	Hidden case	Success	1	0.0741 sec	11.9 KB
TestCase 9	Easy	Hidden case	Success	1	0.1478 sec	11.8 KB

No Comments

QUESTION 8

Correct Answer

Score 0

Difficulty Meter › Multiple Choice

QUESTION DESCRIPTION

On a scale of 1 to 5, with 1 being very easy and 5 being extremely challenging, how would you rate this worksheet?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] 1
- [x] 2
- [x] 3
- [x] 4
- [x] 5

No Comments