You can view this report online at : https://www.hackerrank.com/x/tests/1731388/candidates/57322607/report

| | |
|---|---|
| Full Name: | Instructor |
| Email: | aisha.batool@sse.habib.edu.pk |
| Test Name: | **CS101 - LW10 - Fall23** |
| Taken On: | 22 Oct 2023 19:28:46 PKT |
| Time Taken: | 10 min 5 sec/ 180 min |
| Work Experience: | > 5 years |
| Invited by: | Aisha |
| Skills Score: | |
| Tags Score: | CS101 20/20<br>Python 3 20/20 |

**100%**

**50/50**

scored in **CS101 - LW10 - Fall23** in 10 min 5 sec on 22 Oct 2023 19:28:46 PKT

**Recruiter/Team Comments:**

*No Comments.*

| | Question Description | Time Taken | Score | Status |
|---|---|---|---|---|
| Q1 | Count even up - Recursively > Coding | 5 min 2 sec | 10/ 10 | ✓ |
| Q2 | Tango & Cash - Recursively > Coding | 2 min 27 sec | 10/ 10 | ✓ |
| Q3 | Recursively slow reveal > Coding | 42 sec | 10/ 10 | ✓ |
| Q4 | Wizard's hat full of stars > Coding | 55 sec | 10/ 10 | ✓ |
| Q5 | String search - recursively > Coding | 37 sec | 10/ 10 | ✓ |
| Q6 | Difficulty Meter > Multiple Choice | 6 sec | 0/ 0 | ✓ |

**QUESTION 1**

Correct Answer

Score 10

## Count even up - Recursively > Coding

QUESTION DESCRIPTION

### Problem

Write a *recursive* function named `countup_even` that takes two parameters `a` and `b` and prints all even numbers between `a` and `b` inclusive.

### Sample

```
>>> countup_even(0, 6)
0, 2, 4, 6
>>> countup_even(1, 6)
```

```
2, 4, 6
>>> countup_even(1, 5)
2, 4
>>> countup_even(0, 5)
0, 2, 4
>>> countup_even(1, 1)
>>> countup_even(0, 0)
0
>>> countup_even(2,3)
2
>>> countup_even(3,4)
4
```

## Input

Input `a` and `b` from the console without any prompt.

## Constraints

- `isinstance(a, int)` is `True`
- `isinstance(b, int)` is `True`
- `b >= a` is `True`

INTERVIEWER GUIDELINES

**Solution**

```
a = int(input())
b = int(input())
def countup_even(a,b):
    if a == b:
        if a%2 == 0:
            print(a)
            return
        return
    if a%2 == 0 and a != b-1:
        print(a, end = ', ')
    if a%2 == 0 and a == b-1:
        print(a)
    return countup_even(a+1,b)
```

**CANDIDATE ANSWER**

Language used: **Python 3**

```
1  a = int(input())
2  b = int(input())
3  def countup_even(a,b):
4      if a == b:
5          if a%2 == 0:
6              print(a)
7      elif a%2 == 0 and a != b-1:
8          print(a, end = ', ')
9          countup_even(a+1,b)
10     elif a%2 == 0 and a == b-1:
11         print(a)
12         countup_even(a+1,b)
13     else:
14         countup_even(a+1,b)
15
```

| TESTCASE | DIFFICULTY | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| Testcase 0 | Easy | Sample case | Success | 1 | 0.1543 sec | 11.6 KB |
| Testcase 1 | Easy | Sample case | Success | 1 | 0.0914 sec | 11.7 KB |
| Testcase 2 | Easy | Sample case | Success | 1 | 0.0848 sec | 11.6 KB |
| Testcase 3 | Easy | Sample case | Success | 1 | 0.0961 sec | 11.7 KB |
| Testcase 4 | Medium | Sample case | Success | 1 | 0.1513 sec | 11.6 KB |
| Testcase 5 | Medium | Sample case | Success | 1 | 0.1177 sec | 11.8 KB |
| Testcase 6 | Medium | Hidden case | Success | 2 | 0.1818 sec | 11.5 KB |
| Testcase 7 | Medium | Hidden case | Success | 2 | 0.0702 sec | 11.4 KB |

No Comments

QUESTION 2

Correct Answer

Score 10

## Tango & Cash - Recursively › Coding

Python 3 | CS101

QUESTION DESCRIPTION

Two friends Tango and Cash are trying to see if they understand how to calculate factors correctly. They design a problem in which they will print integers from 1 to N, but print **Tango** if the number is divisible by 3 and print **Cash** if the number is divisible by 5 and print **TangoCash** if the number is divisible by both 3 and 5, if none of the conditions are met the current integer in the sequence will be printed.

**Function Description**
A recursive function **factor_fun** has the following parameter: N (which specifies the number of values)

**Constraints**

- N is a positive integer
- You must write a recursive solution--you may not use *for* or *while* loops, or any other form of iteration.
- You may not reference any global variables.
- You may not use a helper function.

### ▼ Input Format For Custom Testing

The input contains an integer, N.

### ▼ Sample Case 0

**Sample Input For Custom Testing**

```
5
```

**Sample Output**

```
1
2
Tango
4
Cash
```

**Explanation**
The function runs five times and prints Tango when x = 3 and prints cash when x = 5 otherwise it prints the value of x.

### ▼ Sample Case 1

**Sample Input For Custom Testing**

```
16
```

**Sample Output**

```
1
2
Tango
4
Cash
Tango
7
8
Tango
Cash
11
Tango
13
14
TangoCash
16
```

**Explanation**

The function runs for 16 times and prints **Tango** if the value is a multiple of 3, prints **Cash** if the value is a multiple of 5 and prints **TangoCash** when x = 15, otherwise it prints the value of x.

INTERVIEWER GUIDELINES

```
def factor_fun(i):
    if i!=0:
        factor_fun(i-1)
    else:
        return
    output = ""
    if(i%3 == 0):
        output+="Tango"
    if(i%5 == 0):
        output+="Cash"

    if output =="":
        print(i)
    else:
        print(output)
```

**CANDIDATE ANSWER**

Language used: **Python 3**

```
1  def factor_fun(i):
2      if i!=0:
3          factor_fun(i-1)
4
5          output = ""
6          if(i%3 == 0):
7              output+="Tango"
8          if(i%5 == 0):
9              output+="Cash"
10
11         if output =="":
12             print(i)
13         else:
14             print(output)
```

```
15
16
```

| TESTCASE | DIFFICULTY | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|---|---|---|---|---|---|---|
| Testcase 0 | Easy | Sample case | Success | 1 | 0.1231 sec | 11.5 KB |
| Testcase 1 | Easy | Sample case | Success | 1 | 0.0791 sec | 11.4 KB |
| Testcase 2 | Medium | Hidden case | Success | 1 | 0.1383 sec | 11.6 KB |
| Testcase 3 | Medium | Hidden case | Success | 1 | 0.118 sec | 11.7 KB |
| Testcase 4 | Easy | Hidden case | Success | 1 | 0.1171 sec | 11.6 KB |
| Testcase 5 | Medium | Hidden case | Success | 1 | 0.1133 sec | 11.8 KB |
| Testcase 6 | Easy | Hidden case | Success | 1 | 0.0983 sec | 11.6 KB |
| Testcase 7 | Easy | Sample case | Success | 1 | 0.1267 sec | 11.7 KB |
| Testcase 8 | Easy | Sample case | Success | 1 | 0.0825 sec | 11.7 KB |
| Testcase 9 | Easy | Sample case | Success | 1 | 0.0865 sec | 11.6 KB |

No Comments

**QUESTION 3**

Correct Answer

Score 10

## Recursively slow reveal › Coding

QUESTION DESCRIPTION

### Challenge

Write a recursive function `slow_reveal(s)` that takes a string argument `s` and prints out the first letter of `s`, then the first two letters of `s`, then the first three letters of `s`, and so on, until the entire string is revealed.

Note: you may not use while or for loops.

## Sample

```
>>> slow_reveal('Pizza!')
P
Pi
Piz
Pizz
Pizza
Pizza!
```

INTERVIEWER GUIDELINES

```
def helper(i, s):
    if s[:i] == s:
        print(s)
    else:
        print(s[:i])
        helper(i+1, s)


def slow_reveal(s):
    helper(1, s)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def helper(i, s):
    if s[:i] == s:
        print(s)
    else:
        print(s[:i])
        helper(i+1, s)


def slow_reveal(s):
    helper(1, s)
```

| TESTCASE | DIFFICULTY | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|---|---|---|---|---|---|---|
| Testcase 0 | Easy | Sample case | Success | 2 | 0.0647 sec | 11.5 KB |
| Testcase 1 | Easy | Hidden case | Success | 4 | 0.0638 sec | 11.7 KB |
| Testcase 2 | Easy | Hidden case | Success | 4 | 0.0743 sec | 11.6 KB |

No Comments

QUESTION 4

Correct Answer

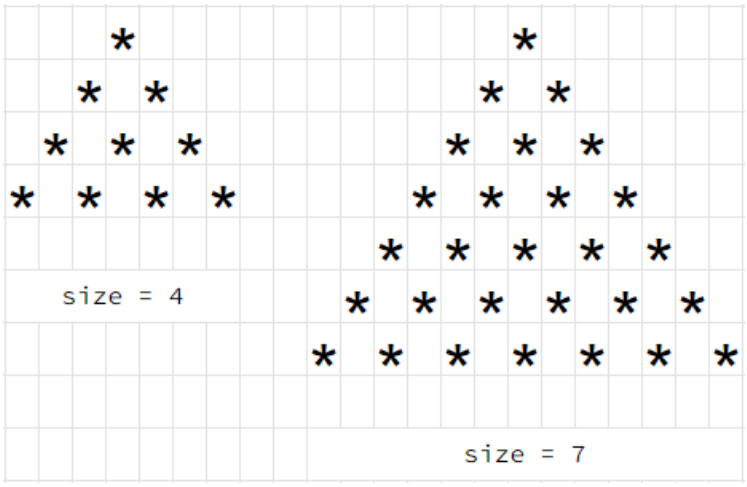
Score 10

## Wizard's hat full of stars › Coding

CS101 Python 3

QUESTION DESCRIPTION

Write a program that draws the shapes of a wizard's hat using space and asterisk characters. The width of the hat is provided as input. See samples below.




**Function Description**

To implement the given task, write the following functions:

1. ***wizard_hat*** that takes one argument: ***size***. The function should draw the shape, as explained above, by calling a helper function, if needed.
2. ***wizard_hat_helper*** (if needed) that accepts any number of arguments, and is called from the starter function.

**Constraints**

- You must use recursion, and possibly string concatenation and repetition, to **print** the repeating pattern--you may not use for or while loops, or any other form of iteration.
- You may not reference global variables.
- Input will be handled by HackerRank--you should not read input yourself.
- *size* is a positive integer.

### ▼ Input Format For Custom Testing

The first and only line contains *size*, the number of stars in the base of the wizard's hat.

### ▼ Sample Case 0

**Sample Input For Custom Testing**

```
4
```

**Sample Output**

```
   *
  * *
 * * *
* * * *
```

**Explanation**

The shape is drawn with four asterisk characters, separated by spaces, as the base of the wizard's hat. The hat has four layers, with each layer above being one star shorter than the previous one. The hat tapers to one star at the very top. First (top) layer has three spaces preceding the first (and only) asterisk, second layer has two spaces, third layer has one space, and fourth (last) layer has no spaces.

### ▼ Sample Case 1

**Sample Input For Custom Testing**

```
7
```

**Sample Output**

```
      *
     * *
    * * *
   * * * *
  * * * * *
 * * * * * *
* * * * * * *
```

**Explanation**

The shape is drawn with seven asterisk characters, separated by spaces, as the base of the wizard's hat. The first layer has six spaces, followed by one asterisk. The second layer has five spaces, followed by two asterisks, separated by one space, and so on.

INTERVIEWER GUIDELINES

```
def wizard_hat(size):
    wizard_hat_helper(size, 1)

def wizard_hat_helper(size, level):
    if level <= size:
        blank = (size - level) * ' '
        stars = level * '* '
        line = blank + stars
        print(line)
```

```
        wizard_hat_helper(size, level + 1)
```

**CANDIDATE ANSWER**

Language used: **Python 3**

```
def wizard_hat(size):
    wizard_hat_helper(size, 1)

def wizard_hat_helper(size, level):
    if level <= size:
        blank = (size - level) * ' '
        stars = level * '* '
        line = blank + stars
        print(line)
        wizard_hat_helper(size, level + 1)
```

| TESTCASE | DIFFICULTY | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|---|---|---|---|---|---|---|
| TestCase 0 | Easy | Sample case | Success | 1 | 0.0729 sec | 11.8 KB |
| TestCase 1 | Easy | Sample case | Success | 1 | 0.084 sec | 11.7 KB |
| TestCase 2 | Easy | Sample case | Success | 1 | 0.1312 sec | 11.7 KB |
| TestCase 3 | Easy | Sample case | Success | 1 | 0.1401 sec | 11.5 KB |
| TestCase 4 | Easy | Sample case | Success | 1 | 0.0757 sec | 11.5 KB |
| TestCase 5 | Easy | Hidden case | Success | 1 | 0.0918 sec | 11.8 KB |
| TestCase 6 | Easy | Hidden case | Success | 1 | 0.0926 sec | 11.5 KB |
| TestCase 7 | Easy | Hidden case | Success | 1 | 0.0979 sec | 11.7 KB |
| TestCase 8 | Easy | Hidden case | Success | 1 | 0.0803 sec | 11.6 KB |
| TestCase 9 | Easy | Hidden case | Success | 1 | 0.076 sec | 11.6 KB |

No Comments

**QUESTION 5**

Correct Answer

Score 10

## String search - recursively › Coding

QUESTION DESCRIPTION

### Background

The string method find returns the position of a sub-string within a string if it is found, -1 otherwise.

### Challenge

Write a function named `my_find` that accepts two parameters, `s` and `subs`. Both parameters will be passed string arguments. The function must **print** the position of the first occurrence of `subs` within `s`, starting from the left of `s`. If `subs` is not found in `s`, the function prints -1. If you are unsure what output to expect for any test case, try running 'find' function of str.

### Note

Your implementation must not use any loops, and must rely on recursion.

## Sample interaction

```
>>> my_find('banana', 'na')
2
>>> my_find('banana', 'no')
-1
>>> my_find('ban', 'na')
-1
```

## Input/Output

Input and output will be handled by HackerRank.

## Hint

Define a *recursive* helper function to call from `my_find`.

INTERVIEWER GUIDELINES

```
# Enter your code here.
def my_find(s, subs):
    helper_find(s, subs, 0)

def helper_find(s, subs, n):
    if len(s) < len(subs):
        print(-1)
    elif s[:len(subs)]==subs:
        print(n)
    else:
        helper_find(s[1:], subs, n + 1)
```

**CANDIDATE ANSWER**

Language used: **Python 3**

```
1  # Enter your code here.
2  def my_find(s, subs):
3      helper_find(s, subs, 0)
4
5  def helper_find(s, subs, n):
6      if len(s) < len(subs):
7          print(-1)
8      elif s[:len(subs)]==subs:
9          print(n)
10     else:
11         helper_find(s[1:], subs, n + 1)
12
```

| TESTCASE | DIFFICULTY | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|---|---|---|---|---|---|---|
| Testcase 0 | Easy | Sample case | Success | 1 | 0.0739 sec | 11.4 KB |
| Testcase 1 | Easy | Hidden case | Success | 1 | 0.0806 sec | 11.5 KB |
| Testcase 2 | Easy | Sample case | Success | 1 | 0.0869 sec | 11.5 KB |
| Testcase 3 | Easy | Sample case | Success | 1 | 0.117 sec | 11.5 KB |
| Testcase 4 | Easy | Hidden case | Success | 1 | 0.1486 sec | 11.6 KB |
| Testcase 5 | Easy | Sample case | Success | 1 | 0.0945 sec | 11.5 KB |
| Testcase 6 | Easy | Hidden case | Success | 2 | 0.1172 sec | 11.5 KB |
| Testcase 7 | Easy | Hidden case | Success | 1 | 0.073 sec | 11.4 KB |

| | | | | | | |
|---|---|---|---|---|---|---|
| Testcase 9 | Easy | Hidden case | Success | 1 | 0.1167 sec | 11.6 KB |

No Comments

**QUESTION 6**

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
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] 5

No Comments

**PDF generated at: 22 Oct 2023 15:06:33 UTC**