You can view this report online at : https://www.hackerrank.com/x/tests/1731412/candidates/57544939/report

Full Name:	Instructor
Email:	aisha.batool@sse.habib.edu.pk
Test Name:	**CS101 - PW11 - Fall23**
Taken On:	29 Oct 2023 22:08:52 PKT
Time Taken:	4 min 31 sec/ 10080 min
Work Experience:	> 5 years
Invited by:	Aisha
Skills Score:	
Tags Score:	CS101 10/10 Easy 20/20 Input 20/20 Iteration 10/10 Log 10/10 Python 3 10/10 Recursion 10/10 cs101 10/10

100%

60/60

scored in **CS101 - PW11 - Fall23** in 4 min 31 sec on 29 Oct 2023 22:08:52 PKT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Sheep Wool Production > **Coding**	1 min 11 sec	10/ 10	✓
Q2	Triangle Number > **Coding**	34 sec	10/ 10	✓
Q3	True Chess Master - Recursive > **Coding**	37 sec	10/ 10	✓
Q4	Silly Lily - Recursive > **Coding**	40 sec	10/ 10	✓
Q5	Chickenville > **Coding**	37 sec	10/ 10	✓
Q6	Binary String > **Coding**	41 sec	10/ 10	✓
Q7	Difficulty Meter > **Multiple Choice**	4 sec	0/ 0	✓

QUESTION 1

✓ Correct Answer

Sheep Wool Production > Coding

Recursion cs101

QUESTION DESCRIPTION

Score 10

Problem Description

We have sheep of two different breeds standing in a line, numbered 1, 2, 3, ...
The odd-numbered sheep (1, 3, ..) produces 0.25 pounds of wool each.
The even-numbered sheep (2, 4, ..) produces 0.5 pounds of wool each.

We want to compute the total weight of wool produced by all sheep.

Complete the **recursive** function `sheepWoolProduction` that takes an integer named `n` as parameter, representing the total number of sheep, and returns the total wool produced.

```
>> sheepWoolProduction(0)
0.0

>> sheepWoolProduction(1)
0.25

>> sheepWoolProduction(2)
0.75
```

Constraints

- `isinstance(n, int)` is `True`
- `n >= 0` is `True`

INTERVIEWER GUIDELINES

Solution

```
#def sheepWoolProduction(sheep):
#    if (sheep == 0):
#        return 0

#    if (sheep%2 != 0):
#        return 0.25 + sheepWoolProduction(sheep-1)
#    else:
#        return 0.5 + sheepWoolProduction(sheep-1)

def sheepWoolProduction(n):
    if (n == 0):
        return 0
    else:
        total_wool = 0

        for index in range(1, n+1):
            if (index % 2 == 0):
                total_wool = total_wool + 0.5
            else:
                total_wool = total_wool + 0.25

        return total_wool
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def sheepWoolProduction(sheep):
   if (sheep == 0):
       return 0

   if (sheep%2 != 0):
       return 0.25 + sheepWoolProduction(sheep-1)
```

```
    else:
        return 0.5 + sheepWoolProduction(sheep-1)

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	1	0.1521 sec	11.7 KB
Testcase 1	Easy	Sample case	✓ Success	1	0.0677 sec	11.5 KB
Testcase 2	Easy	Sample case	✓ Success	1	0.064 sec	11.5 KB
Testcase 3	Easy	Hidden case	✓ Success	1	0.0954 sec	11.5 KB
Testcase 4	Easy	Hidden case	✓ Success	1	0.1304 sec	11.6 KB
Testcase 5	Easy	Hidden case	✓ Success	1	0.1134 sec	11.7 KB
Testcase 6	Easy	Hidden case	✓ Success	1	0.0904 sec	11.6 KB
Testcase 7	Easy	Hidden case	✓ Success	1	0.0564 sec	11.4 KB
Testcase 8	Easy	Hidden case	✓ Success	1	0.0766 sec	11.5 KB
Testcase 9	Easy	Hidden case	✓ Success	1	0.0541 sec	11.7 KB

No Comments

QUESTION 2

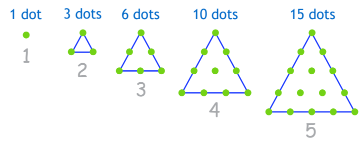
✓ Correct Answer

Score 10

Triangle Number › Coding

QUESTION DESCRIPTION

Problem



Credit: www.mathsisfun.com/algebra/triangular-numbers.html

A triangle number sequence is given as
1, 3, 6, 10, 15, 21, 28, 36, 45, ...
and can be derived geometrically looking at the triangles in the figure on the left.

Write a *recursive* function named `triangle_number` that takes a parameter `n` and returns the corresponding term in the above series.

Sample

```
>>> triangle_number(5)
15
>>> triangle_number(10)
55
>>> triangle_number(1)
1
>>> triangle_number(0)
0
>>> triangle_number(-3)
0
```

Input

Input `n` from the console without any prompt.

Constraints

- `isinstance(n, int)`

INTERVIEWER GUIDELINES

Solution

```
n = int(input())
def triangle_number(n):
    if n <= 0:
        return 0
    return n + triangle_number(n-1)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
# Enter your code here.
n = int(input())
def triangle_number(n):
    if n <= 0:
        return 0
    return n + triangle_number(n-1)

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	2.5	0.0879 sec	11.5 KB
TestCase 1	Easy	Sample case	✓ Success	2.5	0.0695 sec	11.7 KB
TestCase 2	Easy	Sample case	✓ Success	2.5	0.1261 sec	11.4 KB
TestCase 3	Easy	Sample case	✓ Success	2.5	0.0732 sec	11.6 KB

No Comments

QUESTION 3

✓ Correct Answer

Score 10

True Chess Master - Recursive › Coding Easy Input

QUESTION DESCRIPTION

chess.com

Impressed with the inventor of chess, an Indian rajah asks the inventor how they wish to be rewarded. The sly inventor asks for grains of rice to be placed on the chessboard in such a way that one grain is placed on the first square, two grains on the second square, four grains on the third square, eight grains on the fourth square, and so on, doubling the previous number each time, until all 64 squares are filled. The rajah, thinking the reward to be trivial, agrees to the inventor's request. Little did the rajah realize that his granaries would be empty well before he filled the final square.

Function Description

Write a *recursive* function `grains_placed()` that takes the number of squares, `squares`, as parameter and returns `grains`, the total number of grains of rice to be placed on the first `n` squares of the board.

Constraints

- `isinstance(n, int) and n >= 1`

▼ Input Format For Custom Testing

The input only contains `n` on the first line.

The output must be `grains`.

▼ Sample Case 0

Sample Input For Custom Testing

```
1
```

Sample Output

```
1
```

Explanation
The total number of grains on 1 square is 1.

▼ Sample Case 1

Sample Input For Custom Testing

```
5
```

Sample Output

```
31
```

Explanation
The total number of grains on 5 squares is `1 + 2 + 4 + 8 + 16 = 31`.

INTERVIEWER GUIDELINES

Solution

```
# Define function(s).
def grains_placed(squares):
    if squares == 1:
        return 1
    return grains_placed(squares-1) + 2 ** (squares - 1)
```

CANDIDATE ANSWER

Language used: **Python 3**

```

# Define function(s).
def grains_placed(squares):
    if squares == 1:
        return 1
    return grains_placed(squares-1) + 2 ** (squares - 1)

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	1	0.1808 sec	11.7 KB
Testcase 1	Easy	Sample case	✓ Success	1	0.1305 sec	11.8 KB
Testcase 2	Easy	Sample case	✓ Success	1	0.129 sec	11.6 KB
Testcase 3	Easy	Sample case	✓ Success	1	0.1112 sec	11.6 KB
Testcase 4	Easy	Sample case	✓ Success	1	0.0697 sec	11.6 KB
Testcase 5	Easy	Sample case	✓ Success	1	0.0943 sec	11.6 KB
Testcase 6	Easy	Sample case	✓ Success	1	0.0685 sec	11.6 KB
Testcase 7	Easy	Sample case	✓ Success	1	0.1243 sec	11.8 KB
Testcase 8	Easy	Sample case	✓ Success	1	0.1099 sec	11.6 KB
Testcase 9	Easy	Sample case	✓ Success	1	0.0581 sec	11.5 KB

No Comments

QUESTION 4

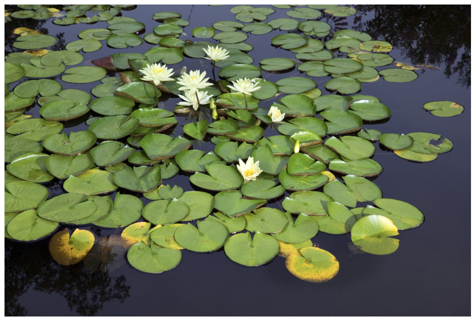
✓ Correct Answer

Score 10

Silly Lily - Recursive › Coding

Easy Input Iteration Log

QUESTION DESCRIPTION



Sciencing

A water lily grows in a pond. Each day, the water lily covers twice as much area as it did the previous day. If the lily starts out as a single pad, it will be 2 pads the next day, 4 pads the next, 8 the next, and so on.

We want to find out how long it will take before a pond containing a single water lily pad is completely covered by lily pads.

Example
Given a pond with area equal to 8 lily pads, it will take 3 days for the pond to be covered. The pond already has 1 lily pad. At the end of Day 1, the pond contains 2 lily pads. On Day 2, it contains 4 and on Day 3, it contains 8 lily pads which completely cover the pond.

Function Description
Write the *recursive* function `days_to_cover()` which takes a parameter `area` indicating the number of lily pads that will completely cover the pond.

Constraints

- `isinstance(area, int) and area >= 1`

▼ Input Format For Custom Testing

The input consists of a single line containing `area`.
The output must contain the number of days needed to cover the pond.

▼ Sample Case 0

Sample Input For Custom Testing

```
1
```

Sample Output

```
0
```

Explanation

`area` is 1, that is the pond will be covered by 1 lily pad. It already contains 1 lily pad so no additional days are required to cover it.

▼ Sample Case 1

Sample Input For Custom Testing

```
10
```

Sample Output

```
4
```

Explanation

`area` is 10, that is the pond will be covered by 10 lily pads. On Days 1, 2, and 3, the pond will have 2, 4, and 8 lily pads respectively. and some part of the pond will still be uncovered. So one more day is required to cover the pond. Note that the pond now has some excess lily pads.

INTERVIEWER GUIDELINES

Solution

```
# Define function.
import math
def days_to_cover(area):
    if area == 1:
        return 0
    return 1 + days_to_cover(math.ceil(area/2))
```

CANDIDATE ANSWER

Language used: **Python 3**

```

import math
def days_to_cover(area):
    if area == 1:
        return 0
    return 1 + days_to_cover(math.ceil(area/2))

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	1	0.0961 sec	11.7 KB
Testcase 1	Easy	Sample case	✓ Success	1	0.1221 sec	11.7 KB
Testcase 2	Easy	Sample case	✓ Success	1	0.0599 sec	11.7 KB
Testcase 3	Easy	Sample case	✓ Success	1	0.1201 sec	11.6 KB
Testcase 4	Easy	Sample case	✓ Success	1	0.1251 sec	11.4 KB
Testcase 5	Easy	Sample case	✓ Success	1	0.1034 sec	11.7 KB
Testcase 6	Easy	Sample case	✓ Success	1	0.0608 sec	11.5 KB

Testcase 7	Easy	Sample case	✓ Success	1	0.0917 sec	11.7 KB
Testcase 8	Easy	Sample case	✓ Success	1	0.0794 sec	11.5 KB
Testcase 9	Easy	Sample case	✓ Success	1	0.0623 sec	11.7 KB

No Comments

QUESTION 5

✓ Correct Answer

Score 10

Chickenville › Coding CS101 Python 3

QUESTION DESCRIPTION

Background

In the online game of *Chickenville*, a farmer must raise a certain number of chickens to pass a given *level*. Difficulty of the game is determined by three values: *start*, *multiplier*, *bonus*.

See the sample cases below to illustrate the mechanics of the game.

Task

Given the values for *start*, *multiplier*, and *bonus*, write a program to calculate the number of chicken that the farmer must raise to pass a *level*.

Function Description

To implement the given task, write the following function:

1. ***level_up*** that takes four argument: ***start***, ***multiplier***, ***bonus***, ***level***. The function should **return** the number of chickens.

Constraints

- You must use recursion to calculate the number of chickens--you may not use for or while loops, or any other form of iteration.
- You may not reference global variables.
- You may not use any helper functions.
- Input and output will be handled by HackerRank--you should not read input or display values yourself.
- *start* and *bonus* are non-negative integers; *multiplier* and *level* are positive integers.

▼ Input Format For Custom Testing

The first line contains *start*, the number of chickens with which the farmer begins the game.
The second line contains *multiplier*, the factor with which to multiply the number of chickens from the previous level.
The third line contains *bonus*, the number to add to the number of chickens from the previous level.
The fourth line contains *level*, which is the desired level for which the number of chickens must be calculated.

▼ Sample Case 0

Sample Input For Custom Testing

```
2
3
5
6
```

Sample Output

```
1091
```

Explanation
The farmer begins the game with two chickens (*start* = 2). To get to the next level (level 2), she must raise 2 x *multiplier* + *bonus* = 2 x 3 + 5 = 11 chickens. For level 3, she must raise 11 x 3 + 5 = 38 chickens. For level 4, 38 x 3 + 5 = 119 chickens. For level 5, 119 x 3 + 5 = 362 chickens. For the final level, *level* = 6, 362 x 3 + 5 = 1091 chickens.

▼ Sample Case 1

Sample Input For Custom Testing

```
1
2
0
10
```

Sample Output

```
512
```

Explanation
The farmer begins the game with one chickens (*start* = 1). To get to the next level (level 2), she must raise 1 x *multiplier* + *bonus* = 1 x 2 + 0 = 2 chickens. For level 3, she must raise 2 x 2 + 0 = 4 chickens. For level 4, 4 x 2 + 0 = 8 chickens. For level 5, 8 x 2 + 0 = 16 chickens. For level 6, 16 x 2 + 0 = 32 chickens. For level 7, 32 x 2 + 0 = 64 chickens. For level 8, 64 x 2 + 0 = 128 chickens. For level 9, 128 x 2 + 0 = 256 chickens. For the final level, *level* = 10, 256 x 2 + 0 = 512 chickens.

INTERVIEWER GUIDELINES

```
def level_up(start, multiplier, bonus, level):
    if level == 1:
        return start
    else:
        return multiplier * level_up(start, multiplier, bonus, level - 1)
+ bonus
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def level_up(start, multiplier, bonus, level):
    if level == 1:
        return start
    else:
        return multiplier * level_up(start, multiplier, bonus, level - 1) +
bonus
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	1	0.084 sec	11.7 KB
TestCase 1	Easy	Sample case	✓ Success	1	0.0977 sec	11.7 KB
TestCase 2	Easy	Sample case	✓ Success	1	0.1094 sec	11.8 KB
TestCase 3	Easy	Sample case	✓ Success	1	0.0742 sec	11.6 KB
TestCase 4	Easy	Sample case	✓ Success	1	0.1022 sec	11.7 KB
TestCase 5	Easy	Hidden case	✓ Success	1	0.1163 sec	11.7 KB

TestCase 6	Easy	Hidden case	✓ Success	1	0.1378 sec	11.7 KB
TestCase 7	Easy	Hidden case	✓ Success	1	0.0892 sec	11.7 KB
TestCase 8	Easy	Hidden case	✓ Success	1	0.0606 sec	11.6 KB
TestCase 9	Easy	Hidden case	✓ Success	1	0.1082 sec	11.4 KB

No Comments

QUESTION 6

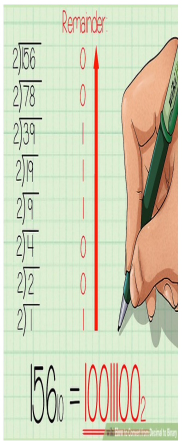
✓ Correct Answer

Score 10

Binary String › Coding

QUESTION DESCRIPTION

Problem



Credit: wikiHow

You are helping your friends with converting numbers to binary. The process is to divide the number repeatedly by 2 until the quotient becomes 1. When the remainder from each step is written in reverse order, the required binary number is obtained. This is illustrated in the image on the left.

Write a *recursive* function named `binary_string` that takes an `int` parameter `n` and returns the binary representation of `n` as a `str`.

Sample

```
>>> binary_string(0)
'0'
>>> binary_string(156)
'10011100'
>>> binary_string(-3)
'-11'
```

Input

Input `n` from the console without any prompt.

Constraints

- `isinstance(n, int)`

INTERVIEWER GUIDELINES

Solution

```
n = int(input())
def binary_string(n):
    '''binary_string(int) -> str

    Return binary representation of n as a string.
    '''
    if n < 0:
        return '-' + binary_string(-n)
    if n <= 1:
        return str(n)
    return binary_string(n//2) + str(n%2)
```

CANDIDATE ANSWER

Language used: **Python 3**

```python
# Enter your code here.
n = int(input())
def binary_string(n):
    '''binary_string(int) -> str

    Return binary representation of n as a string.
    '''
    if n < 0:
        return '-' + binary_string(-n)
    if n <= 1:
        return str(n)
    return binary_string(n//2) + str(n%2)

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	1	0.0812 sec	11.7 KB
TestCase 1	Easy	Sample case	✓ Success	1	0.0775 sec	11.6 KB
TestCase 2	Easy	Sample case	✓ Success	1	0.0996 sec	11.5 KB
TestCase 3	Easy	Sample case	✓ Success	1	0.1007 sec	11.7 KB
TestCase 4	Easy	Sample case	✓ Success	1	0.0807 sec	11.8 KB
TestCase 5	Easy	Sample case	✓ Success	1	0.1282 sec	11.4 KB
Testcase 6	Easy	Sample case	✓ Success	1	0.0844 sec	11.6 KB
Testcase 7	Easy	Sample case	✓ Success	1	0.07 sec	11.5 KB
Testcase 8	Easy	Sample case	✓ Success	1	0.07 sec	11.6 KB
Testcase 9	Easy	Sample case	✓ Success	1	0.1314 sec	11.6 KB

No Comments

QUESTION 7

✓ 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)

✓ [● 1]

✓ ○ 2

✓ ○ 3

✓ ○ 4

✓ ○ 5

No Comments

PDF generated at: 29 Oct 2023 17:13:58 UTC