Worksheet: Recursion

CS 101 Algorithmic Problem Solving

Fall 2023

Name(s): _____

HU ID *(e.g., xy01042)*: _____

1. Brownies

Elijah has a young daughter that he drops off at the local daycare before going to work at his bakery. Yesterday, Elijah's bakery had N brownies left over and Elijah brought them home. As the brownies can last for another day, he thought he should drop off some of the brownies at his daughter's daycare. He knows there are a total of M kids at the daycare including his daughter and he will only bring the highest amount of brownies that can be divided equally among the kids and any leftover brownies will be given to the homeless shelter nearby.

You are to make a **recursive** function Leftovers, with integer parameters N and M, that returns the number of brownies that will be given to the homeless shelter.

Given values of N and M, give the output of the Leftovers(N,M).

In the Pseudocode section, make sure to print the function output as print(functionname(parameters))

Constraints

- $N, M \in \mathbb{Z}$
- $1 \leq N, M \leq 50$

Interaction

The input comprises a single line containing 2 space-separated integers denoting the values of N and M respectively.

The output must contain a single number denoting the number of brownies given to the homeless shelter.

Sample

Input	Output
17 8	1
23 23	0

In the first case, $(N, M) = (17, 8)$; Elijah takes 16 brownies to the daycare, 2 for each of the 8 kids, and the remaining 1 brownie to the homeless shelter.

In the second case, $(N, M) = (23, 23)$; Elijah takes 23 brownies to the daycare, 1 for each kid, and none to the homeless shelter.

Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
12 11	1
43 12	7
12 15	12

Pseudocode

```
def Leftovers(N,M):
    if N < M:
        return N
    else:
        return Leftovers(N-M,M)
print(Leftovers(N,M))
```

Dry Run
Using any one of the inputs provided in the Exercise section above, dry run your pseudocode in the space below.

```
N = 43, M = 12
1. Leftovers(43, 12):

- N is not less than M, so the function goes to the else statement and returns
Leftovers(43 - 12, 12) = Leftovers(31, 12).

2. Leftovers(31, 12):

- N is not less than M, so the function goes to the else statement and returns
Leftovers(31 - 12, 12) = Leftovers(19, 12).

3. Leftovers(19, 12):

- N is not less than M, so the function goes to the else statement and returns
Leftovers(19 - 12, 12) = Leftovers(7, 12).

4. Leftovers(7, 12):

- N is less than M, so it returns the value of N, which is 7, which is the expected output.

Therefore, for the inputs N = 43 and M = 12, the function Leftovers will return 7.
```

Problem Identification
Briefly explain the underlying problem you identified in the above question that led you to your solution.
Input: N,M
Output: Calculate the remainder after dividing the number of brownies among number of children by using recursive calls

2. Wool Production

A farm has sheeps of two different breeds, standing in a line, numbered 1, 2, 3, and so

on. The sheeps at odd-numbered positions are one breed, and they produce 0.25 pounds of wool each and sheeps at even-numbered positions are the other breed, and they produce 0.5 pounds of wool each.

You are to write a **recursive** function called SheepProd, that takes in an integer parameter N, and returns the total amount of wool the sheeps produce.

Given the value of N, determine the output of SheepProd(N).

In the Pseudocode section, make sure to print the function output as print(functionname(parameters))

Constraints

- $N \in \mathbb{Z}$
- $0 \leq N \leq 10^5$

Interaction

The input comprises a single line containing an integer representing the value of N.

The output must contain a single number denoting the amount of wool produced by all sheep.

Sample

Input	Output
0	0.0
2	0.75

In the first case, $N = 0$; there are 0 sheep and they produce 0 pounds of wool.

In the second case, $N = 2$; there are 2 sheep, the first produces 0.25 pounds of wool and the second produces 0.5 pounds, and in total they produce 0.75 pounds of wool.

Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
4	1.5
33	12.25
1253	469.75

Pseudocode

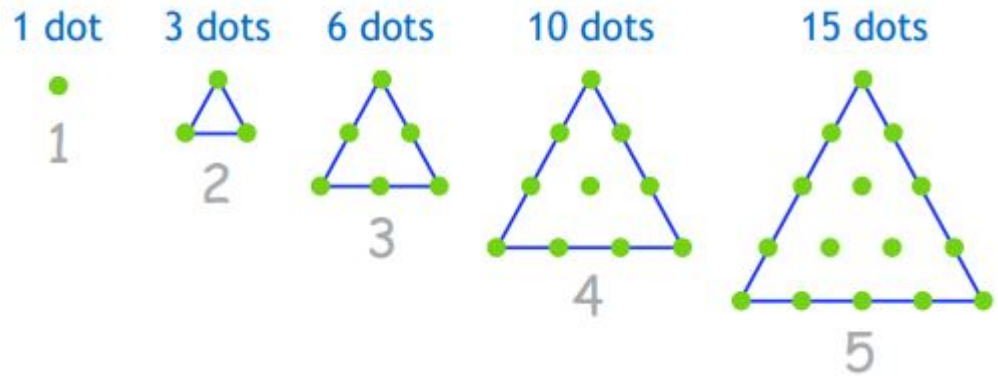
```
def SheepProd(N):
    if N == 0:
        return 0.0
    elif N == 1:
        return 0.25
    else:
        if N % 2 == 0:
            return SheepProd(N - 1) + 0.5
        else:
            return SheepProd(N - 1) + 0.25
print(SheepProd(N))
```

Dry Run
Using any two of the inputs provided in the Exercise section above, dry run your pseudocode in the space below.

```
1. SheepProd(4):

- Since N = 4 is even, the function goes to the condition
if N % 2 == 0 and returns SheepProd(3) + 0.5.

2. SheepProd(3):

- Since N = 3 is odd, the function goes to the condition
else and returns SheepProd(2) + 0.25.

3. SheepProd(2):

- N = 2 is not equal to 0 or 1, so the function goes
to the condition if N % 2 == 0 and returns SheepProd(1) + 0.5.

4. SheepProd(1):

- N = 1 is not equal to 0, so the function goes to the condition
 elif N == 1 and returns 0.25.

Now, we backtrack through the calls:

- SheepProd(1) = 0.25
- SheepProd(2) = SheepProd(1) + 0.5 = 0.25 + 0.5 = 0.75
- SheepProd(3) = SheepProd(2) + 0.25 = 0.75 + 0.25 = 1.0
- SheepProd(4) = SheepProd(3) + 0.5 = 1.0 + 0.5 = 1.5

So, for N = 4, the SheepProd function will return 1.5 - which
is the expected output!
```

Problem Identification
Briefly explain the underlying problem you identified in the above question that led you to your solution.
Input: N
Output: Calculate and return total wool production using recursive calls and adding the appropriate amounts.

3. Triangle Numbers



Credit: Math Is Fun

A triangle number sequence is given as 1, 3, 6, 10... and can be derived geomterically by looking at the triangles above. Your friend introduced you to this sequence and a the first thought in your head was to try and build an algorithm to generate terms of this sequence. You've been introduced to recursion recently and suspect that it will be greatly useful for this problem.

You are to write a **recursive** function TriangleNum that takes an integer parameter N and returns the corresponding term of the sequence.

Given the value of N, determine the output of TriangleNum(N).

In the Pseudocode section, make sure to print the function output as print(functionname(parameters))

Constraints

- $N \in \mathbb{Z}$
- $1 \leq N \leq 10^5$

Interaction

The input comprises a single line containing an integer representing the value of N.

The output must contain a single number denoting the Nth term of the triangle number sequence.

Sample

Input	Output
3	6
5	15

In the first case, $N = 3$; the triangle comprises of 6 dots.

In the second case, $N = 5$; the triangle comprises of 15 dots.

Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
7	28
11	66
9	45

Pseudocode

```
def TriangleNum(N):
    if N == 1:
        return 1
    else:
        return N + TriangleNum(N-1)
print(TriangleNum(N))
```

Dry Run

Using any one of the inputs provided in the Exercise section above, dry run your pseudocode in the space below.

1. TriangleNum(7): Since N is not equal to 1, the function goes to the else statement and returns 7 + TriangleNum(6).

2. TriangleNum(6): Since N is not equal to 1, the function goes to the else statement and returns 6 + TriangleNum(5).

3. TriangleNum(5): Since N is not equal to 1, the function goes to the else statement and returns 5 + TriangleNum(4).

4. TriangleNum(4): Since N is not equal to 1, the function goes to the else statement and returns 4 + TriangleNum(3).

5. TriangleNum(3): Since N is not equal to 1, the function goes to the else statement and returns 3 + TriangleNum(2).

6. TriangleNum(2): Since N is not equal to 1, the function goes to the else statement and returns 2 + TriangleNum(1).

7. TriangleNum(1): Since N is equal to 1, the function returns 1.

Now, we backtrace through the calls:

- TriangleNum(1) = 1
- TriangleNum(2) = 2 + 1 = 3
- TriangleNum(3) = 3 + 3 = 6
- TriangleNum(4) = 4 + 6 = 10
- TriangleNum(5) = 5 + 10 = 15
- TriangleNum(6) = 6 + 15 = 21
- TriangleNum(7) = 7 + 21 = 28

So, for N = 7, the TriangleNum function will return 28 which is the expected output!

Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.
Input: N

Output: Compute Nth term of the triangle number sequence by using recursive calls and return the appropriate output.

4. Facht Auriel's Riddle

You've bought an entrance ticket to Facht Auriel's Castle and your ticket number is N. The castle houses an ancient blade of immense power that can only be obtained by those wise enough to crack the riddles created by Facht Auriel himself. An entrance ticket gives you one shot at the riddles. However, nobody has been able to solve any of the riddles yet.
Finally, your turn to enter the castle arrives. You enter a large room in the center of which is a statue of Facht Auriel, with his two hands extended out. On one of his hands, there's a pen, a paper, and a riddle that says:

"You are the product of those that came before you and failed. Scribble who you are on this parchment and await Facht Auriel's judgement.".

Confused by the riddle at first, you sit down to think. After some time, you realize the answer must be the product of the ticket numbers of all those who came before you.

You have decided to make a **recursive** function called facht that takes an integer parameter N denoting your ticket number and determines the answer of the riddle.

Constraints

- $N \in \mathbb{Z}$
- $2 \leq N \leq 10^5$

Interaction

The input comprises of a single integer denoting the value of N.

The output must be a single integer denoting the answer to the riddle.

Sample

Input	Output
4	6
3	2

In the first case, $N = 4$, so the answer to the riddle is $1 * 2 * 3 = 6$.

In the second case, $N = 3$, so the answer to the riddle is $1 * 2 = 2$.

Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
12	39916800
5	24
9	40320

Pseudocode

```
1 def facht(N):
2     if N == 2:
3         return 1
4     else:
5         return (N-1)*facht(N-1)
6 print(facht(N))
```

Dry Run

Using any one of the inputs provided in the Exercise section above, dry run your pseudocode in the space below.

```
1. facht(5): Since N is not equal to 2, the function goes to the else statement
and returns (5-1)*facht(4).
2. facht(4): Since N is not equal to 2, the function goes to the else statement
and returns (4-1)*facht(3).
3. facht(3): Since N is not equal to 2, the function goes to the else statement
and returns (3-1)*facht(2).
4. facht(2): Since N is equal to 2, the function returns 1.

Now, we backtrack through the calls:

- facht(2) = 1
- facht(3) = (3-1)*facht(2) = 2*1 = 2
- facht(4) = (4-1)*facht(3) = 3*2 = 6
- facht(5) = (5-1)*facht(4) = 4*6 = 24

So, for N = 5, the function facht(N) will return 24, which is the expected output!
```

Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.

Input: N

Output: Product of all numbers from 1 to $N - 1$ by using recursive calls.

5. Another Riddle

Having succeeded with Facht Auriel's riddle, you stand around waiting for someone to bring you your shiny new sword. However, to your surprise, a paper drops at your feet. You pick it up to find yet another riddle that reads:

"You walk an uncharted path, oh young traveler. Knock at the door of greatness as many times as the tally of your great name and the blade will be yours to wield."

You take a moment to process that you've gotten further than anyone else before sitting down once again to think on the riddle. You look around and as you look at the door from whence you came, the solution strikes you. The riddle wants you to knock at the door as many times as the sum of the digits in your answer to the last riddle.

You've decided to make a **recursive** function for this as well, called facht2, which takes an integer parameter A and returns the answer of the riddle.

Constraints

- $A \in \mathbb{Z}$
- $1 \leq A \leq 10^5$

Interaction

The input comprises of a single integer denoting the value of A.

The output must be a single integer denoting the answer to the second riddle.

Sample

Input	Output
6	6
24	6

In the first case, $A = 6$, so the answer to the riddle is 6.

In the second case, $A = 24$, so the answer to the riddle is $2 + 4 = 6$.

Proposed Solution

```
1  def facht2(A):
2    if A == 0:
3        return 0
4    else:
5        return A//10 + facht2(A%10)
6
7  print(facht2(A))
```

Dry Run

Using the inputs provided in the Sample section above, dry run the proposed code solution below.

```
1. facht2(6): Since A is not equal to 0, the function goes to the else
statement and returns 6//10 + facht2(6%10).

2. facht2(6%10): 6 % 10 is 6, so the function returns 6//10 + facht2(6).
Here, the function will keep calling itself with the same value, leading
to an infinite loop.
```

```
The error in the code is that the recursive call doesn't ensure that
the base case is eventually reached. The function keeps calling itself
with the same value, causing an infinite loop.
```

Error Identification
Briefly explain the errors you identified in the proposed code solution. Mention the line number and the errors in each line.

1. **Line 5**: The recursive call does not ensure that base case is reached, it ends up in infinite recursive calls. Value of A is not decrementing correctly when recursive call is made.

Correct Solution
Rewrite the lines of code you mentioned above with their errors corrected.

```
1 def facht2(A):
2   if A == 0:
3       return 0
4   else:
5       return A%10 + facht2(A//10)
6 print(facht2(A))
```

Rough Work