Worksheet: Recursion

CS 101 Algorithmic Problem Solving

Fall 2023

Name(s): _____

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

1. Count up

Given a number `n`, Ali wants to print the numbers starting from `0` till `n`. Write a recursive function `print_numbers` to help him out. The function takes input `n` and prints the numbers as output.

In the Pseudocode section, make sure to call the function as `functionname(parameters)`

Constraints

- `isinstance(n, int)`
- `1 <= n <= 100`

Interaction

The input comprises a single line containing an integer denoting the value of `n`.

The output must contain numbers counting up till `n`.

Sample

Input	Output
4	0
	1
	2
	3
	4

In the first case, the counting started from 0 and ended at 4

Exercise

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

Input	Output
2	0
	1
	2

Pseudocode

```
def print_numbers(n):
    if n < 0:
        return
    print_numbers(n - 1)
    print(n)
print_numbers(n)
```

Dry Run

Using the input provided in the Exercise section above, dry run your pseudocode in the space below.

```
n = 3
Step 1: print_numbers(2) calls print_numbers(1), since 2 is not less than 0
Step 2: print_numbers(1) calls print_numbers(0), since 1 is not less than 0
Step 3: print_numbers(0) calls print_numbers(-1), since 0 is not less than 0
Step 4: print_numbers(-1) returns immediately, as -1 is less than 0
Step 5: Execution continues with the print(n) statements:
print(0) is executed.
print(1) is executed.
print(2) is executed.
Output:
0
1
2 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: Print numbers from `0` to `n` by calling a function recursively till a base case (`n < 0`) is reached.

2. Star Pattern

Given a number `n` design a recursive function named `print_stars` that prints a staircase of stars with `n` steps. The number of stars increment with each step, starting from 1 star on the first step. The function takes input `n` and prints the staircase pattern as output.

In the Pseudocode section, make sure to call the function as `functionname(parameters)`

Constraints

- `isinstance(n, int)`
- `1 <= n <= 100`

Interaction

The input comprises a single line containing n.

The output must contain a staircase of n steps.

Sample

Input	Output
3	*
	**
	***

In the first case, there are 3 levels and each level has number of stars corresponding to that level.

Exercise

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

Input	Output
4	*
	**
	***
	****

Pseudocode

```
def print_stars(n):
    if n<0:
        return
    print_stars(n-1)
    print("*"*n)
print_stars(n)
```

Dry Run

Using the input provided in the Exercise section above, dry run your pseudocode in the space below.

1. print_stars(4) is called.
2. Since (4) is not less than (0), the function calls print_stars(3).
3. Since (3) is not less than (0), the function calls print_stars(2).
4. Since (2) is not less than (0), the function calls print_stars(1).
5. Since (1) is not less than (0), the function calls print_stars(0).
6. Since (0) is not less than (0), the function calls print_stars(-1).

```
7. Now, since (-1) is less than (0), the function returns without further recursion.
8. Moving back up the recursive calls, print("*" * 1) is executed.
9. Moving back up the recursive calls, print("*" * 2) is executed.
10. Moving back up the recursive calls, print("*" * 3) is executed.
11. Moving back up the recursive calls, print("*" * 4) is executed.

The output of this function call will be:


*
**
***
****
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: Print a staircase pattern of stars by defining and then calling a function recursively till a base case (`n < 0`) is reached.

3. Count odd down

Given a number `n`, Ali wants to print all odd numbers starting from `n` till `1` (inclusive). Write a recursive function named `print_odd` that counts down the odd numbers starting from `n` to `1` and prints them as output.

In the Pseudocode section, make sure to call the function as `functionname(parameters)`

Constraints

- `isinstance(n, int)`
- `1 <= n <= 100`

Interaction

The input comprises a single line containing an integer denoting the value of `n`.

The output has odd numbers printed starting from `n` to 1.

Sample

Input	Output
8	7
	5
	3
	1

In the first case, the countdown started from 8 and only printed odd numbers down to and including 1.

Exercise

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

Input	Output
5	5
	3
	1

Pseudocode

```
def print_odd(n):
    if n < 0:
        return
    if n % 2 !=0:
        print(n)
    print_odd(n-1)
print_odd(n)
```

Dry Run

Using the input provided in the Exercise section above, dry run your pseudocode in the space below.

1. print_odd(5) is called.
2. Since (5) is not less than (0) and (5) is an odd number, the function prints (5).
3. The function calls print_odd(4).
4. Since (4) is not less than (0) and (4) is not an odd number, the function doesn't print anything and proceeds to the next recursive call.

```
5. The function calls print_odd(3).
6. Since (3) is not less than (0) and (3) is an odd number, the function prints (3).
7. The function calls print_odd(2).
8. Since (2) is not less than (0) and (2) is not an odd number, the function
doesn't print anything and proceeds to the next recursive call.
9. The function calls print_odd(1).
10. Since (1) is not less than (0) and (1) is an odd number, the function prints (1).
11. The function calls print_odd(0).
12. Since (0) is not less than (0), the function doesn't print anything and
 immediately returns without further recursion.

The output of this function call will be:
5
3
1
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: Print all odd numbers starting from `n` to `1` inclusive by defining and calling a function recursively till a base case is reached.

Rough Work