Homework 1

## Tower of Hanoi

Most materials are from Wikipedia:

The tower of Hanoi is shown in the figure.



**The disks are put in the left rod (source rod) and they should be put in the right rod (target rod).** The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.
3. No larger disk may be placed on top of a smaller disk.

With 3 disks, the puzzle can be solved in 7 moves.

A recursive solution for m disks is shown below:

1. Move m−1 disks from the source to the spare rod. This leaves the disk m as a top disk on the source rod.
2. Move the disk m from the source to the target rod.
3. Move the m − 1 disks that we have just placed on the spare, from the spare to the target rod by the same general solving procedure, so they are placed on top of the disk m without violating the rules.

### First Problem

Write a recursive solution for the Hanoi Problem. You need to write a function with four arguments:

1. The first argument is the number of disks.
2. The second is the name of source rod.
3. The third is the name of spare rod.
4. The fourth is the name of target rod.

There is no return value for this function. Use “print” to show the solution. Your solution should be optimal.

Test case:

>>> hanoi(1, 'A', 'B', 'C')

A->C

>>> hanoi(2, 'A', 'B', 'C')

A->B

A->C

B->C

>>> hanoi(3, 'A', 'B', 'C')

A->C

A->B

C->B

A->C

B->A

B->C

A->C

### Second Problem

There is an advanced problem of Hanoi, which adds one more rule:

All moves must be between adjacent rods (i.e. given rods A, B, C, one cannot move directly between rods A and C).

Write a recursive solution for this problem. Similar to first problem, you need to write a function with four arguments:

1. The first argument is the number of disks.
2. The second is the name of source rod.
3. The third is the name of spare rod.
4. The fourth is the name of target rod.

Your solution should be optimal. (Tips: If the number of the disks is n, the number of least moves is 3 \*\* n - 1.)

Test case:

>>> hanoi\_plus(1, 'A', 'B', 'C')

A->B

B->C

>>> hanoi\_plus(2, 'A', 'B', 'C')

A->B

B->C

A->B

C->B

B->A

B->C

A->B

B->C

>>> hanoi\_plus(3, 'A', 'B', 'C')

A->B

B->C

A->B

C->B

B->A

B->C

A->B

B->C

A->B

C->B

B->A

C->B

A->B

B->C

B->A

C->B

B->A

B->C

A->B

B->C

A->B

C->B

B->A

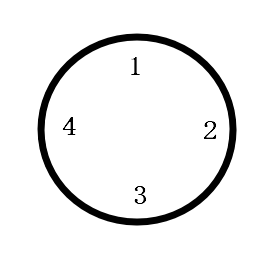
B->C

A->B

B->C

## The Josephus Problem

Several people are numbered from 1 to n. They stand as a circle clockwise (If n == 4, the situation is shown in figure):



Start counting (the number begins from 1) clockwise from the first person, if someone announces an even number, he/she should leave the circle. The counting continues until there is only one person in circle.

For example, if n == 4, No.1 announces 1, and then No.2 announces 2. 2 is even so No.2 is out. Then No.3 announces 3. No.4 announces 4 then out. Now circle remains No.1 and No.3. No.1 announces 5 and No.3 announces 6. No.3 is out. So the answer is 1 because only No.1 is in circle.

### First Problem

Write a function with one argument (i.e. the number of people, which must be a positive integer) to simulate this process until there is only one person left. **Return** the id of the remain person.

Test cases:

>>> print(circle(1))

1

>>> print(circle(2)) # Sequence of the person who is counting: 1 (2)

1

>>> print(circle(3)) # Sequence of the person who is counting: 1 (2) 3 (1)

3

>>> print(circle(4)) # Sequence of the person who is counting: 1 (2) 3 (4) 1 (3)

1

Tips: You can use “list” in Python to simulate. Use “pop” to delete a number with specific index.

### Second Problem

There are some formulas to give the answer of this problem directly:

Use recursive function to find the remain person. You don’t need to simulate again.

Test cases are same as the first problem.

### Third Problem

There is another formula to give the answer of this problem directly:

Use this formula to find the remain person. You don’t need to simulate again.

Test cases are same as the first problem.

**Submit your code to Canvas before the deadline.**

**DDL 2022-11-11 23:59:59**

**No Latter is allowed!!!!**