CS 602
Algorithm Design and Implementation

Assignment 1

[Question 1] Power Modulo

The RSA algorithm has been demonstrated in class, where its encryption and decryption both require the power modulo operation. Given three positive integers *m*, *k* and *n*, the power modulo operation calculates m^k modulus *n*, denoted as $r = m^k \bmod n$, where $r \in \mathbb{Z}$ and $0 \leq r < n$.

Please write a program to implement the power modulo operation. Test inputs begin with a number indicating the number of lines below. For each row below, there are three numbers: *m*, *k* and *n*. For each line, please output the result $r = m^k \bmod n$ on a single line using `print`.

Sample code is provided in the file **A1Q1.py**. You need to modify the function `power_modulo`. *m*, *k*, *n* are all integers between 1 and 2^{31}-1.

There will be 6 sets of test cases with 1 mark each, 2 additional marks are awarded if your code is implemented using recursion given that your code passes all 6 sets of test cases. The remaining 2 marks are awarded with the correct justification of the time complexity of your algorithm in comments, which usually begin and end with three apostrophe signs.

Sample Input

```
3
1 6 7
2 10 10
602 10000019 1000000007
```

Sample Output

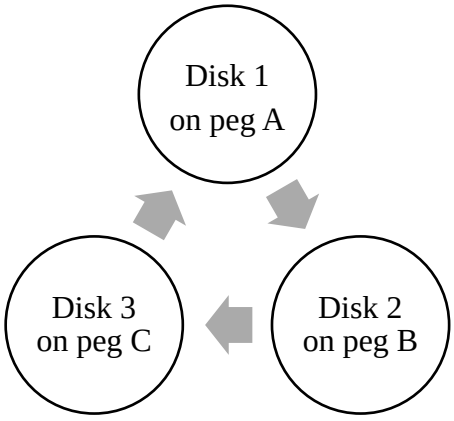
```
1
4
699467277
```

[Question 2] Tower of Hanoi

The Tower of Hanoi is a mathematical puzzle invented by the French mathematician Édouard Lucas in 1883. There is a story about an Indian temple in Kashi Vishwanath which contains a large room with three time-worn posts in it, surrounded by 64 golden disks. Brahmin priests, acting out the command of an ancient prophecy, have been moving these disks in accordance with the immutable rules of Brahma since that time. The puzzle configuration and rules are as follows:

- There are *n* disks and three pegs: A, B, C. The *n* disks are different in size, and all disks are initially placed on peg A that their sizes increase from top to bottom.
- Only one disk can be moved at a time, from the top of a peg to an empty peg or to a peg with larger disk than itself on top.
- The goal is to move all *n* disks from peg A to either peg B or peg C.

It is easy to prove that **the minimum number of moves** needed to complete the puzzle with *n* disks is 2^n-1 using recursion.



A simple algorithm for the minimum number of moves is to move the smallest disk (disk 1) on odd moves in a circular manner ABCABC… (clockwise, as shown on left) or ACBACB… (counterclockwise); and make the only possible move not involving disk 1 on even moves. Given a state of *n* disks placed on pegs A, B and C with their sizes increasing from top to bottom, what is the minimum number of moves to restore all disks on the same peg according to the simple algorithm above? For example, with 3 disks at a state that disk 1 (the smallest disk) on peg A, disk 2 (the medium disk) on peg B and disk 3 (the largest disk) on peg C, the disks can be restored on peg C with 2 moves, first move disk 2 from peg B to peg C, and then move disk1 from peg A to peg C counterclockwise.

The algorithm itself gives 2 options to move disk 1, either clockwise or counterclockwise. There are also 2 options to start restoring, either we start with an odd move (move disk 1) or start with an even move (move other disks than disk 1). That gives us a total of 4 options, however, not all options lead to the state of all disks on the same peg. The table below describes all four options with the state above, but the second and third columns do not restore all disks. The first column gives us a solution to restore all disks with 5 moves on peg A, while the fourth column gives us a solution to restore all disks with only 2 moves on peg C, which is also the solution described in the paragraph above. Since we are looking for a solution with minimum number of moves, the solution in the fourth column is the one we desire. Some state may not give any solution, for example, if disk 1 and 3 are on peg A, disk 2 is on peg B, there is no solution according to the algorithm above.

	Odd move first clockwise			*Odd move first counterclockwise*			*Even move first clockwise*			*Even move first counterclockwise*		
	A	B	C	A	B	C	A	B	C	A	B	C
initial	1	2	3	1	2	3	1	2	3	1	2	3
1		1				1			2			2
	∅	2	3	∅	2	3	1	∅	3	1	∅	3
												1
2		1				1			2			2
	3	2	∅	2	∅	3	∅	1	3	∅	∅	3
3	3	2	1	2	1	3	2	1	3			
4	2					2			1			
	3	∅	1	∅	1	3	2	∅	3			
	1											
5	2					2			1			
	3	∅	∅	1	∅	3	∅	2	3			
6				1	2	3	1	2	3			

Please write a program to determine which option should be adopted and how many moves are needed to restore all disks. Test inputs begin with a number indicating the number of test cases below, and each test case is described by a line describing the state with disks on pegs A, B, C separated by comma, and then separated by space between disks in ascending order. For each test case, output (i) the peg which all disks will be restored at, (ii) number of moves to restore all disks, and (iii) the option to be adopted if the number of moves is not zero, it must be one of the four options: “odd clockwise”, “odd counterclockwise”, “even clockwise” and “even counterclockwise” Output “impossible” if no option leads to the state of all disks on the same peg.

Sample code is provided in the file **A1Q2.py**. You need to modify the function `tower_hanoi`. The total number of disks is between 3 and 64.

There will be 6 sets of test cases with 1 mark each, 2 additional marks are awarded if your code is implemented using recursion given that your code passes all 6 sets of test cases. The remaining 2 marks are awarded with the correct justification of the time complexity of your algorithm in comments, which usually begin and end with three apostrophe signs, same as Q1.

Sample Input

```
9
 , , 1 2 3
 1, 2, 3
 , 2, 1 3
 , 3, 1 2
 1 4, 3, 2
 2 4, 3, 1
 1 4, 2 5 8 9, 3 6 7 10
 1 2 7 8 11 12 15 16 17 22 25 26, 3 14 23 28, 4 5 6 9 10 13 18 19 20 21 24 27
 19 20 21 24, 1 2 3 6 7 8 9 10 11 14 15 16 17 22 25 28 29 30, 4 5 12 13 18 23 26 27
```

Sample Output

```
C 0
C 2 even counterclockwise
impossible
B 3 odd clockwise
A 5 odd counterclockwise
impossible
C 405 odd clockwise
B 77714235 odd counterclockwise
B 111286296 even counterclockwise
```

[Question 3] Asymptotic Analysis

Using any of the techniques for solving recurrence in class, provide an asymptotic bound for T(n) for each of the following recurrences:

i. $T(n) = 6\,T(n/3) + n \lg n$
ii. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$

Please write your answer in a word document A1Q3.docx, please keep your answer within 1 page for this question.

Running python skeleton with sample input:

1. Open "Anaconda Prompt"
2. Go to the directory where you put the file **A1Q1.py** and **A1Q1.in**, using command **`cd`**
3. Run command **python A1Q1.py < A1Q1.in**
4. You may want to create a test input called **my_own_test.in** to design a test case for your own program, the command would then be **python A1Q1.py < my_own_test.in**
5. Same applies to Question 2, so you may run **python A1Q2.py < A1Q2.in**