OUR EXPECTATIONS:

1. We expect you to come up with a **simple console application** in the **language of your choice**. There is no need for a UI, or a web application.
2. With this exercise we are expecting to see how you write production ready code by focusing on:
   - **Simple and modular design**.
   - Clean code practices using OO / functional programming
   - **Unit test** case coverage.
   - Handling boundary conditions
   - Code styles etc.
3. Please stay **within the boundaries** defined in the problem. Avoid over-thinking and over-engineering. Clearly state your assumptions wherever needed in the code or in the README file.
4. Submit your solution in a **public GitHub / Bitbucket repo.**

OTHER EXPECTATIONS (nice to have):

1. Please mention the **setup instructions** and **how to run the program** in the README file.
2. Divide the problem statement in smaller tasks / features and have small and atomic commits in your repo.

PROBLEM STATEMENT - SNAKES & LADDERS:

You are required to create a program, which simulates the playing of a **Snakes & Ladders game**.

Board: There are 100 cells on a Snakes & Ladders board from 01, 02... all the way to 100. However, your starting position is **00**, which is **outside the board**.

Assumptions:

- Assume that the game has **4 snakes** and **4 ladders** of varying lengths dispersed on the board.
- You can choose to initialize / define **where** these snakes and ladders will be on the board and how **long** each of them are. For e.g, a ladder could be at cell 07 and it could take you ahead to 33. Or a snake could be at 87 and bring you back to 32. It's your choice.

Sample ladder position:

Ladder foot	Ladder top
07	33
37	85
51	72

Sample snake position:

Snake head	Snake tail
36	19
65	35
87	32

Inputs and Outputs for your program:

The **input** to your program will be **any number between 1 to 6** (...the numbers on a dice...), and your **current position** on the board

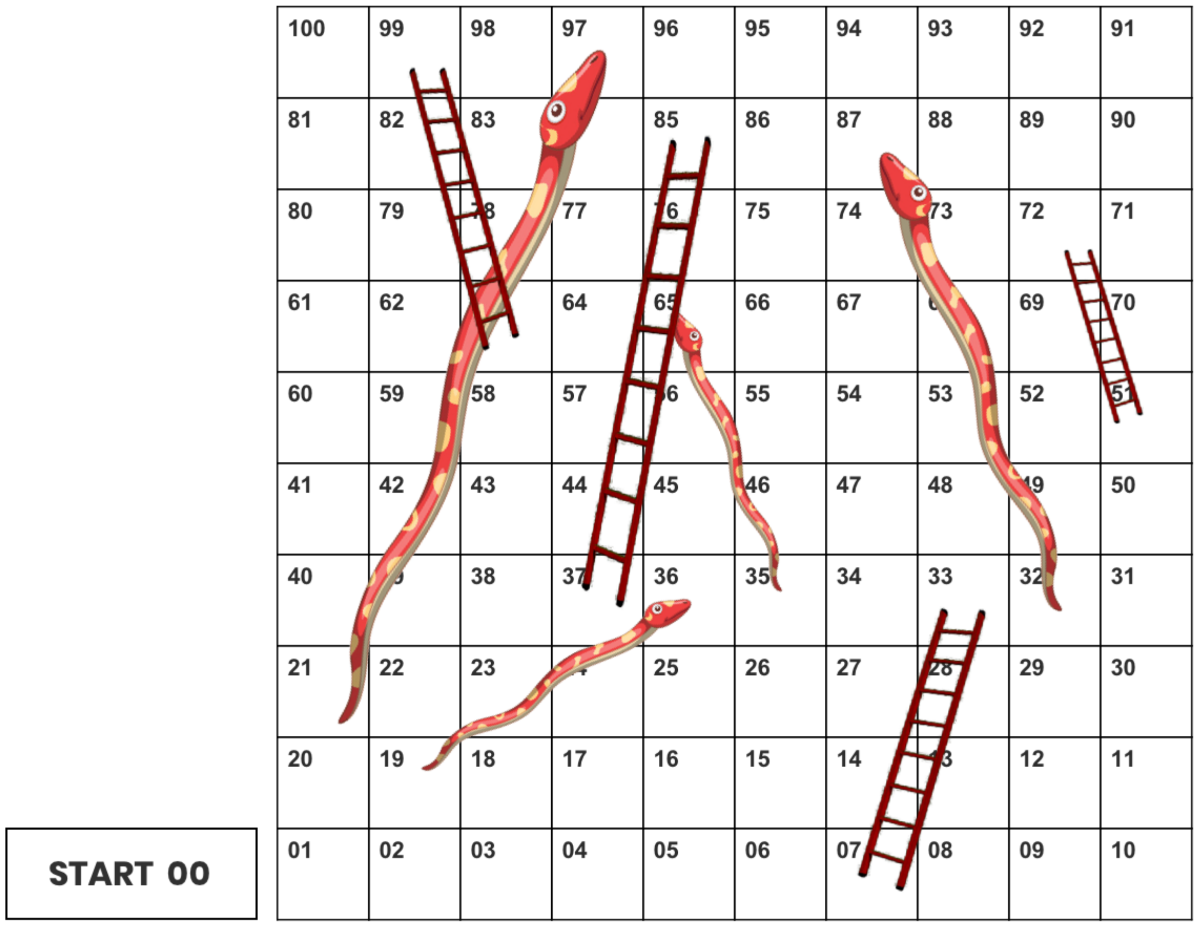
The **output** of your program will be your **new position** on the board.

End of game: The game ends as soon as you reach exactly 100. ***However, if the output comes out to be more than 100, then the piece will remain in the current position.*** See examples below.

Examples:

Input - Current position, Dice outcome	**Output** - New position
04, **5**	New position: **9**
34, 3	**85** (*assuming that there is a **ladder from 37 to 85***)
83, 4	**32** (*assuming that there is a **snake from 87 to 32***)
96, 5	**96** (*since 96+5 is more than 100*)
99, 1	**Yay!! You won!!** (and exit the program)

Sample board illustration:



100
99
98
97
96
95
94
93
92
91
81
82
83
85
86
87
88
89
90
80
79
77
75
74
73
72
71
61
62
64
65
66
67
69
70
60
59
58
57
55
54
53
52
41
42
43
44
45
46
47
48
50
40
38
37
36
35
34
33
32
31
21
22
23
25
26
27
28
29
30
20
19
18
17
16
15
14
12
11
01
02
03
04
05
06
07
08
09
10
START 00