S1-A. Snakes and Ladders

Time limit	3500 ms
Memory limit	128 MB

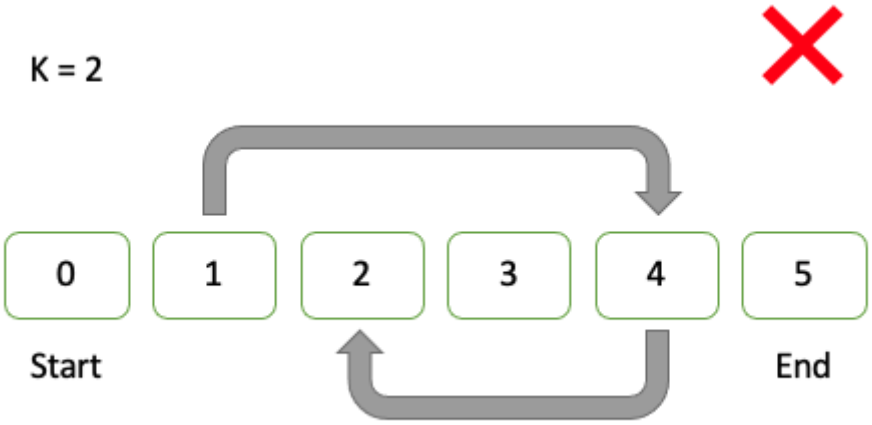
Description

The Dengklek Board Game Festival is starting soon! You, as a fan of Snakes and Ladders, would like to contribute by creating and showing off Snake and Ladders puzzle in that festival. There are several variants of Snake and Ladders rules, you would like to create a Snakes and Ladders with the following characteristics:

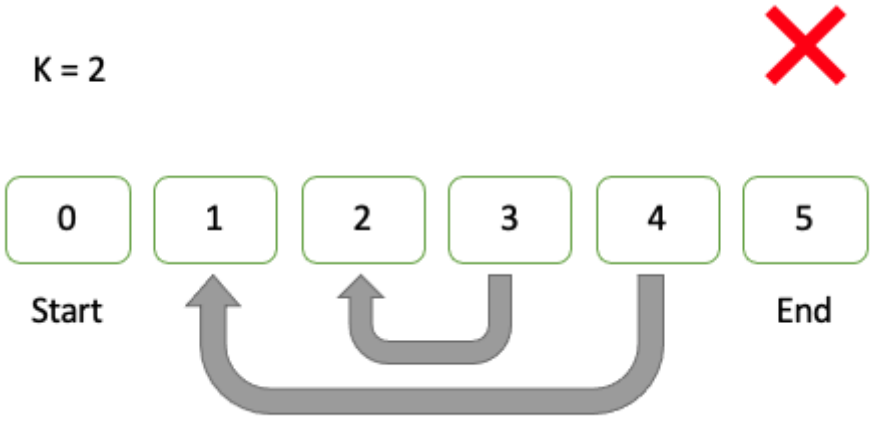
- The Snakes and Ladders board consist of $N + 2$ tiles numbered from 0 to $N + 1$, with tile 0 as the starting tile and tile $N + 1$ as the destination tile.
- The game is played with a K-sided dice. In one turn, a player can throw the dice and get a value i where $1 \leq i \leq K$. If, before throwing the dice, the player is at position p, then the player's position will change to position $\min(p + i, N + 1)$.
- There are several snakes and/or ladders in a board. Both snakes and ladders can be represented as a pair of integers s and t. If, after throwing a dice, a player move to position s, then the player's position will change to position t. Snake always brings you to a lower tile or in other words $t < s$. On the other hand, ladder always bring you to a higher tile or in other words $t > s$.
- There must be no snakes or ladders at the starting tile (tile 0) or at the destination tile (tile $N + 1$). In other words, for every snake and ladder (s, t), the following condition holds $1 \leq s, t \leq N$.
- There can not be more than one snake or ladder in a tile. In other words, every tile is one of the following 5 types.
  - Empty tile
  - A tile with a starting ladder.
  - A tile with an ending ladder.
  - A tile with a starting snake.
  - A tile with an ending snake.
- There exists a sequence of dice such that the player can reach the destination tile by following that sequence.

Followings are some examples of valid and invalid configurations.

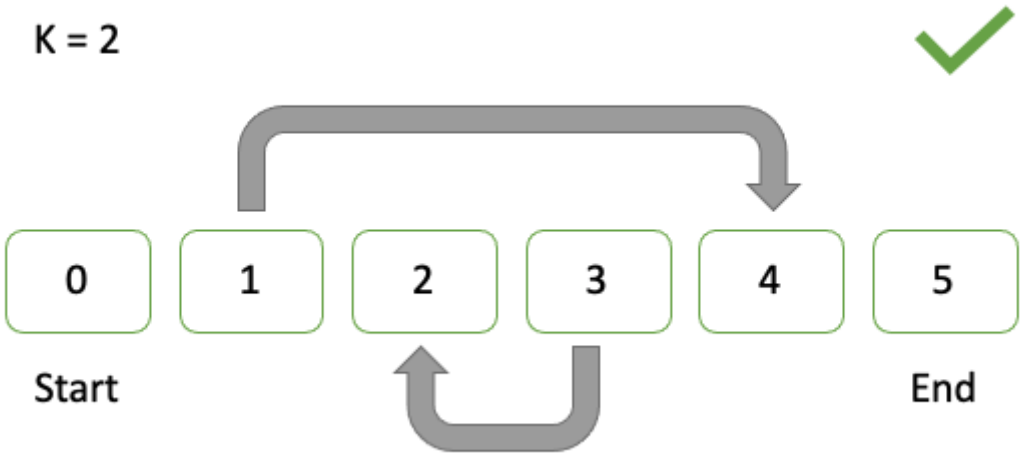
- The following configuration is an **invalid** example because at tile 4, there are an ending ladder and a starting snake.



- The following configuration is an **invalid** example because there is no sequence of 2-sided dice that can reach tile 5.



- The following is an example of a **valid** configuration



You wonder how many valid board configurations are there for an integer N and K modulo a prime number M.

Task

You must implement the following function.

```
int count_valid_board(int N, int K, int M)
```

- `N`: An integer that represents the number of tiles other than the starting tile and destination tile.
- `K`: An integer that represents the dice sides.
- `M`: An integer that represents the modulo. It's guaranteed that M is a prime number.
- This function must return an integer representing the number of valid board configurations with $N + 2$ tiles and K-sided dice modulo M.

Example

If given N = 2, K = 1, and $M = 10^9 + 7$, then function `count_valid_board(N, K, M)` must return 2. The first valid board has neither ladder nor snake. The second valid board has one ladder that starts at tile 1 and ends at tile 2. Other than that, there is no other valid board configurations.

If given N = 3, K = 1, and $M = 10^9 + 7$, then function `count_valid_board(N, K)` must return 4. The first valid board has neither ladder nor snake. The second, third, and fourth valid board has one ladder with position (1, 2), (1, 3), and (2, 3) respectively. Other than that, there is no other valid board configurations.

If given N = 4, K = 2, and $M = 10^9 + 7$, then function `count_valid_board(N, K)` must return 23.

Subtasks

For all subtasks:

- $1 \leq K \leq N \leq 64$
- $2 \leq M \leq 2 \times 10^9$, where M is a prime number.

Subtask 1 (9 Points)

- $N \leq 8$

Subtask 2 (16 Points)

- $N \leq 18$
- $K = 1$

Subtask 3 (27 Points)

- $K = 1$

Subtask 4 (33 Points)

- $N \leq 25$

Subtask 5 (15 Points)

- No additional constraints

Sample Grader

The provided sample grader read input with the following format:

- A single line consists of 3 integers N, K, and M.

The provided sample grader output with the following format:

- A single line consists of an integer A, the value returned by the function `count_valid_board`