# 3. Chutes

**Program Name: Chutes.java Input File: chutes.dat**

You have been contacted by a game company to write a game engine that can play Chutes and Ladders for any arbitrary 8 by 8 game board. The path along the board from 1 to 64 is shown in the grid to the right.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 64 | 63 | 62 | 61 | 60 | 59 | 58 | 57 |
| 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
| 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 |
| 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 32 | 31 | 30 | 29 | 28 | 27 | 26 | 25 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

The rules of the game are pretty simple:

- All players begin play in square one, the lower left corner of the grid.
- Each player then, in order, takes a turn rolling a 6 sided die with the numbers 1 through 6 on the sides.
- The player then, beginning from square number one, advances his token the number of squares that corresponds to the number he rolled on the die.

After a player has moved the number of spaces that are indicated on the die, he will have moved to a space that has exactly one of three possibilities:

- If a player lands on a space that contains the bottom of a ladder (referred to as "landing on a ladder"), he moves up the ladder to a new space as described below.
- If he lands on a space with the top of a chute (referred to as "landing on a chute"), he moves down the chute to a new space as described below.
- If the new space he lands on is neither a chute nor a ladder, he stays at that position until his next turn.

Other rules are:

- If case 1 or case 2 above moves him to another ladder or chute, he performs the same action as listed above, and continues doing so until he gets to a space that is neither a chute nor a ladder. At that point in the game, he is in state 3 above and he stays at that position until his next turn.
- Players rotate taking their turns beginning with player A as described below.
- The game is won when a player rolls a number that will move him to (or past) the box labeled 64.
- There is guaranteed to not be a chute in 64, as that would prevent the game from being solvable.
- There will also never be a chute or ladder starting from 1.

To simulate the dice rolls, you should construct an object of the type `java.util.Random`. This class allows you to specify the seed for the random number generator. For a given seed, the order of the random numbers is always the same. You should also use the `nextInt` method to generate the rolls.

## Input

The first line will contain a single integer `n` that indicates the number of games to follow. For each game,

- The first line contains a single integer `p` that indicates the number of players in the game.
- The next line will contain the seed, a `long` integer.
- The following line will contain 2 integers `c` and `d` separated by a space. The first integer `c` is the number of chutes; the second `d` is the number of ladders.
    - Next are `c` lines containing an integer pair for each chute; the first integer is where a chute begins and the second is where the player ends up at the end of the chute.
    - Then there are `d` lines containing an integer pair for each ladder; the first integer is where the ladder begins, and the second is where the player ends up at the top of the ladder.
    - The integer values for the chute and ladder pairs correspond to the values in the grid above.

**(continued on next page)**

# 3. Chutes (cont.)

**Output**
Each player in a game is given an uppercase character, with the first player being `A`, the second being `B`, etc. For each game you should output a single line that reads `Player X wins after Y rolls!` where *X* stands for the winner of the game, and *Y* is the total number of rolls that happened overall in the game, including the winning roll. If for example there are two players, and player `A` rolled 6 times and player `B` rolled 5, and player `A`'s sixth roll won the game, then the string would read `Player A wins after 11 rolls!`

**Example Input File**
```
2
2
3735928559
4 4
27 10
10 6
57 41
34 14
14 37
17 49
50 64
22 37
3
4206243583
3 4
61 45
60 45
45 37
17 48
4 54
11 39
40 56
```

**Note: these are the random numbers generated until the final roll for each seed in the input file:**
```
3735928559
2 3 5 6 4 2 6 4 4 4 5 4 2 4 3 5 2 4 4 2 5 1 4 5 5
4206243583
2 2 5 6 2 2 1 2 6 5 4 3 6 4 6 4 1 4 1 5 3 1 2 4 2 6 6 2 5 6 6 3
```

**Example Output to Screen**
```
Player A wins after 25 rolls!
Player B wins after 32 rolls!
```