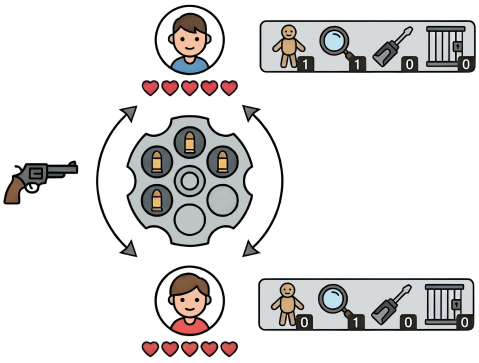
Problem G
Revolver Roulette
Time limit: 15 seconds

This is an interactive problem. Remember to flush the output buffer after every print. To flush your output, you can use:

- `fflush(stdout)` or `cout.flush()` in C/C++;
- `System.out.flush()` in Java and Kotlin;
- `sys.stdout.flush()` in Python.



The concept of the game, illustrated by Nano Banana Pro.

You are participating in a strategic turn-based game known as *Revolver Roulette*.

The game begins with a revolver loaded with exactly 6 bullets. Each bullet is either *live* or *blank*, independently with equal probability. Both players always know the current counts of live and blank bullets remaining in the cylinder, but the exact order is unknown.

Each player starts with 5 health points (HP) and 2 items. If a player's HP drops to 0 or below, they are eliminated, and the other player wins. Players take turns performing actions.

Turn Procedure

During their turn, the active player may perform any number of the following actions:

- Use any item.
- Perform any shooting action.

The turn ends only when a shooting action causes the turn to pass to the other player, or some player is eliminated.

Shooting Actions

The active player must choose one of the following shooting actions:

- `Fire at Opponent`: The active player fires the next bullet at the other player. If it is live, the other player loses 1 HP. Regardless of the bullet type, the turn passes to the other player.
- `Fire at Self`: The active player fires the next bullet at themselves. If it is live, the active player loses 1 HP, and the turn passes to the other player. Otherwise, if it is a blank, the active player retains the turn and continues performing actions.

Items

When an item is used, it is consumed, and the active player retains the turn, regardless of the item type. Items are visible to both players. When a player obtains a new item (including each of the initial items), it is chosen uniformly at random from the following four types:

- `Dummy`: Fire the next bullet at a training dummy. This consumes the bullet and is not considered a shooting action.
- `Magnifier`: Peek at the next bullet to reveal whether the next bullet is live or blank. Both players learn the result.

- `Converter`: Flip the type of the next bullet (live ↔ blank). Both players observe the updated counts of live and blank bullets.
- `Cage`: Gain a one-time effect that prevents the turn from ending on the next shooting action that would normally end it. Specifically:
  - If the player later performs `Fire at Opponent` (regardless of whether the bullet is live or blank), the effect applies and the turn does not end.
  - If the player later performs `Fire at Self` and the bullet is live, the effect applies and the turn does not end (the player still loses 1 HP, but retains the turn).

  The effect is consumed immediately after it applies once. At most one `Cage` item may be used per turn.

Reloading

When the cylinder is empty, 6 new bullets are immediately loaded (each independently live or blank with equal probability). Both players are informed of the new counts of live and blank bullets. Additionally, each player is resupplied up to 2 items (a player with k items receives $(2 - k)$ new items, each independently chosen uniformly at random from the above four types).

Goal

You are the player who takes the first turn, and your goal is to win the game by eliminating your opponent.

Input

The first line contains two integers n and w, indicating that you need to win at least w out of n games to get accepted. There are only three tests:

- Sample test: $n = 1$ and $w = 0$. (0%)
- Secret test 1: $n = 2\,000$ and $w = 1\,000$. (50%)
- Secret test 2: $n = 20\,000$ and $w = 11\,000$. (55%)

It can be shown that, when both players play optimally, the player who takes the first turn wins approximately 56.89% of the games.

Interaction Protocol

Game Start

Your solution acts as the player who takes the first turn against the interactor. At the start of each game, the interactor prints an integer g:

- If $g = 0$, all games in the current run are complete. Your solution must terminate immediately.
- Otherwise, g is the current game index ($1 \le g \le n$).

Due to technical issues, your solution is executed **twice** on each test:

- **The interactor can decide whether to restart your solution** by giving $g = 0$ at any time.

You will receive the same n and w for each run in a test. Therefore, n is **not** always equal to the number of games in the current run. It is guaranteed that each game $1, \ldots, n$ is played exactly once across the two runs. Apart from this, your solution can safely ignore the run twice condition.

Game State

Then, the interactor prints the game state line containing 13 non-negative integers:

$$t \quad l \quad b \quad h_1 \quad d_1 \quad m_1 \quad c_1 \quad k_1 \quad h_2 \quad d_2 \quad m_2 \quad c_2 \quad k_2$$

- $t \in \{0, 1, 2\}$: turn status.
- l, b ($1 \le l + b \le 6$): the current counts of live and blank bullets remaining in the cylinder.
- h_1 ($0 \le h_1 \le 5$): your current HP.
- d_1, m_1, c_1, k_1 ($0 \le d_1 + m_1 + c_1 + k_1 \le 2$): the number of `Dummy`, `Magnifier`, `Converter`, and `Cage` items you currently hold.
- h_2 ($0 \le h_2 \le 5$): your opponent's current HP.
- d_2, m_2, c_2, k_2 ($0 \le d_2 + m_2 + c_2 + k_2 \le 2$): the number of `Dummy`, `Magnifier`, `Converter`, and `Cage` items your opponent holds.

Case $t = 0$ (Game Over)

The current game has ended. The interactor proceeds to the next game (printing the new g).

Case $t = 1$ (Your Turn)

You must print an integer to choose an action:

1. `Fire at Opponent`.
2. `Fire at Self`.
3. Use a `Dummy` item.
4. Use a `Magnifier` item.
5. Use a `Converter` item.
6. Use a `Cage` item.

If the action is invalid (e.g., using an item you don't have, or using a `Cage` item more than once in the same turn), the game state remains unchanged, the interactor prints 0 and the same state line (13 integers) again. Although it remains your turn, you must make another valid action to proceed with the game.

Otherwise, the action is effective and the interactor prints a result integer:

- For actions 1–5: Returns 1 if the next bullet before the action is live, 2 if it is blank.
- For action 6: Returns 1 (no bullet info revealed).

The interactor then prints the updated game state line.

Case $t = 2$ (Opponent's Turn)

The interactor simulates your opponent's action and prints two lines:

- The action ID $a \in \{1, \ldots, 6\}$.
- The result integer (1 or 2 as described above).

Your opponent never makes an illegal action. Immediately afterward, the interactor prints the next game state line (13 integers).

For each round of the game, the game simulator uses a pseudorandom number generator with a fixed random seed. Also, the opponent in the interactor employs a deterministic strategy. The decision of restarting your solution is independent of the interaction procedure. Therefore, the same sequence of actions always leads to the same game state.

A testing tool is provided to help you develop and test your solution.

Read **Sample 1, Pass 1** **Write**

```
1 0
1
1 3 3 5 0 0 0 2 5 1 0 1 0
```

```
6
```

```
1
1 3 3 5 0 0 0 1 5 1 0 1 0
```

```
1
```

```
1
1 2 3 5 0 0 0 1 4 1 0 1 0
```

```
6
```

```
0
1 2 3 5 0 0 0 1 4 1 0 1 0
```

```
2
```

```
2
1 2 2 5 0 0 0 1 4 1 0 1 0
```

```
1
```

```
1
2 1 2 5 0 0 0 1 3 1 0 1 0
5
2
2 2 1 5 0 0 0 1 3 1 0 0 0
1
1
1 1 1 4 0 0 0 1 3 1 0 0 0
```

```
1
```

```
1
2 0 1 4 0 0 0 1 2 1 0 0 0
3
2
2 2 4 4 0 1 0 1 2 0 1 0 1
2
1
1 1 4 4 0 1 0 1 1 0 1 0 1
```

```
4
```

```
1
1 1 4 4 0 0 0 1 1 0 1 0 1
```

```
1
```

```
1
0 0 4 4 0 0 0 1 0 0 1 0 1
0
```

Read **Sample 1, Pass 2** **Write**

```
1 0
0
```