A general “subtraction game” is specified by a number \( m \in \mathbb{N} \) and a finite set \( S \subseteq \mathbb{N} \) where \( m \) is called the initial game configuration and \( S \) constitutes the set of allowed moves. It is assumed that \( 1 \in S \). A subtraction game \( \langle m, S \rangle \) is a two player game with the following rules:

- Initial setup: A pile of \( m \) coins is put on a table.
- Player 1 is the player that makes the first move.
- At every move, a player removes a number \( k \) of coins from the table so that \( k \in S \).
- The winner is the player that removes the last coin from the pile.

**Task.** Any subtraction game has a winning strategy for one of the two players. Write a program that takes as input the specification \( \langle m, S \rangle \) and returns the player that has a winning strategy (i.e., your program should return either “player 1” or “player 2”).

**Extra Extra Credit:** write an interactive program that allows a human to play with the computer a subtraction game. The program should offer to the user to select \( m \) and \( S \) as well as specify who is player 1 and player 2 (computer or human). If the computer has a winning strategy your program should always win over the user. If the user has a winning strategy, your program should always win if the user makes one wrong move.

In order to solve this problem you should consider the following:

- For each game \( \langle m, S \rangle \) there is a total of \( m + 1 \) game configurations corresponding to the number of coins left in the pile, i.e., elements of the set \( \{0, \ldots, m\} \).
- The game configuration graph of each game \( \langle m, S \rangle \) is a directed graph with \( m + 1 \) vertices \( V = \{0, \ldots, m\} \) and a set of edges \( E \) such that \( (i, j) \in E \) if and only if \( i - j \in S \).
- Vertices can be labeled as P-positions or N-positions. A P-position means that the player that moved previously has a winning strategy or won the game, whereas an N-position means that the player that moves next has a winning strategy. Clearly the vertex 0 is a P-position. Labeling of other vertices in the graph is possible using the following two rules:
  1. A vertex is a P-position if all vertices that it can reach in one step are N-positions.
  2. A vertex is an N-position if it can reach a vertex that is a P-position in one step.
- Player 1 has a winning strategy if and only if vertex \( m \) is labeled as N-position.

**Implementation hint.** For the implementation, consider reusing your graph data-structure implemented in homework 5. Additionally, you may find the reverse graph and/or reverse BFS very useful.