Programming project 2

Due: see Canvas
Group work: You may work in groups of 1-3. Include all group members when submitting to Gradescope.

Write a Python class, *GameOfNim*, that defines the rules of the Game of Nim:

1. Two players take turns removing objects from distinct heaps or rows
2. In each turn, a player must remove at least one object from the same row
3. **The winner is the player that takes the last object(s).**[1]

As in project 1, you will not be implementing any game search algorithm. You only need to implement the class with game rules such that it can be used with an adversarial search algorithm to play optimally against any opponent.

- The class must extend class Game in the games.py code.
- Represent the state by a list which represents the number of objects in each pile/row. E.g., [5, 3, 1] represents 5 objects in the first row, 3 in the second, and 1 in the third row.
- An action in this game is removing a certain number of objects from one pile. Represent an action by a 2-tuple (r, n) where r represents the row number (start counting from 0 for convenience as Python uses 0-based indexing) and n represents the number of objects to remove. E.g., (1,2) means remove 2 objects from row with index 1 (the second row).

The class should be usable in a main function to play a game between the computer (using the minmax with alpha-beta pruning algorithm) and a human (query_player). See Canvas for the main. The main should produce output like so (user's input is shown in red):

```
[0, 5, 3, 1]
[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 1), (2, 2), (2, 3), (3, 1)]
GameState(to_move='MIN', utility=0, board=[0, 2, 3, 1], moves=[(1, 1), (1, 2),
(2, 1), (2, 2), (2, 3), (3, 1)])

move: (1, 3)
resulting state:
board:  [0, 2, 3, 1]
available moves: [(1, 1), (1, 2), (2, 1), (2, 2), (2, 3), (3, 1)]

Your move? (1,2)
resulting state:
board:  [0, 0, 3, 1]
```

[1] Note that this is a variant of the game in the movie where the player who picked the last object *lost* the game

```
move: (2, 2)
resulting state:
board:  [0, 0, 1, 1]
available moves: [(2, 1), (3, 1)]

Your move? (2,1)
resulting state:
board:  [0, 0, 0, 1]

move: (3, 1)
resulting state:
board:  [0, 0, 0, 0]
MAX won the game
```

Hints:

- The class will be similar to class TicTacToe in games.py. (Note that only the structure of the code is similar as the rules of the TicTacToe and Nim are very different.) Specifically, your class should
    1. Extend abstract class Game

        ```
        class GameOfNim(Game):
        ```

    2. have a **constructor** which takes the initial board position and creates the initial GameState. Note that a GameState includes all valid moves for that state. For example, if the board position= `[0, 2, 3, 1]`, the valid moves are: `[(1, 1), (1, 2), (2, 1), (2, 2), (2, 3), (3, 1)]`
    3. have a method **result(state, move)** that returns the new state reached from the given state and the given move. Assume the move is a valid move. Note that the state for a multiplayer game also includes the player whose turn it is to play
    4. have a method **actions(state)** that returns a list of valid actions in the given state. This is easy if you generate the list of valid moves when a child state is created.
    5. have a method **terminal_test(state)** that returns True if the given state represents the end of a game
    6. have a method **utility(state, player)** that returns +1 if MAX wins, -1 if MIN wins (the "names" of the players don't matter as long as they are distinct)
    7. have a method **to_move(state)** that returns the player whose turn it is to move. The default implementation in abstract class Game should be sufficient.
- It is recommended to implement the methods in the order above and test each method individually before starting the next method.
- For debugging, you can also override Game.play_game() to print the current state and player (as I did to produce the sample output above)

Submit:

Submission will be via Gradescope. Upload a single file with your class. The name of the file must be `game_of_nim.py`. The name of the class should be `GameOfNim`.
Gradescope will run a few unit tests automatically and show you the results. You can submit multiple times.