# CSC236 Problem Set 2

## Question 5.

(a) I'll call the player who takes the first turn player 1.

Base Case: Let two piles have 1 and 2 pebbles respectively.

Player 1 just need to take 1 pebble from the pile that has two 2 pebbles. Since the player may not remove pebbles from both piles during the same turn, no matter which pile player 2 choose, player 2 must remove at least one pebble, lefting only a pile with 1 pebble left. Thus, player 1 can win by remove the left pebble.

I've proved the base case is true.

Induction Step. Let $m, n \in \mathbb{N}$. $m \neq n$. Let two piles contain $m$ and $n$ pebbles respectively.

Induction Hypothesis. Let $p \in \mathbb{N}$, $2 \leq p < \max(m, n)$ Assume player I will win if two piles contain $p$ pebbles and less then $p$ pebbles, call $q$, respectively.

WTP. Player 1 can win if two piles contain $m$ and $n$ pebles respectively.

Player 1 just need to take $\max((n-m), (m-n))$ pebble from pile 1 (contains $n$ pebbles) or pile 2 (contains $m$ pebbles) at first turn, respectively, making the pebbles in both piles equal. No matter which pile player 2 chooses, the pile player 2 choose will less then the other pile, both piles have pebble less than $n$, since the player must remove at least 1 pebble. According to I.H., take $p = \max($the number of pebbles in pile I, the number of pebbles in pile 2$)$, gives play 1 must win.

I've proved the induction step.

Therefore. If the game starts with two piles having different numbers of piles, then there's a strategy that guarantees a win for the player who takes the first turn. ■

(b).

```python
def p1_win_strategy(n: int, m: int) -> list:
    """
    Develop the winning strategy for player 1, which player 1 should keep the number of two piles the same each turn.

    :param n: the number of pebbles in a pile, called pile 1
    :param m: the number of pebbles in another pile, called pile 2
    :return: the rest of the pebbles in each pile as a list

    Precondition:
    - The first player who takes the first turn is called Player 1
    - m < n
    - Both m and n are natural number
    - The player must remove at least one pebble
    - The player may remove up to every pebble from one pile
    - The player may not remove pebbles from both piles during the same turn
    - The player may choose a different pile in a different turn
    """
    if n > m and m !=0:
        # when pile 1 contains more pebbles and pebbles in the other pile is not 0, then make two piles the same pebbles
        # by subtracting (n - m) pebbles from pile 1
        n -= (n - m)
    elif n > m and m == 0:
        # when pile 1 contains more pebbles and pebbles in the other pile is 0, then make two piles the same pebbles
        # by subtracting n pebbles from pile 1 (every pebbles), which player 1 will win
        n -= n
    elif n < m and m != 0:
        # when pile 2 contains more pebbles and pebbles in the other pile is not 0, then make two piles the same pebbles
        # by subtracting (m - n) pebbles from pile 1
        m -= (m - n)
    else:
        # when pile 2 contains more pebbles and pebbles in the other pile is 0, then make two piles the same pebbles
        # by subtracting m pebbles from pile 2, which player 2 will win
        m -= m
    return [n, m]
def p2_move_strategy(n: int, m: int) -> list:
    """
    Impletment a random move from player 2

    :param n: the number of pebbles in a pile, called pile 1
    :param m: the number of pebbles in another pile, called pile 2
    :return: the rest of the pebbles in each pile as a list

    Precondition:
    - The first player who takes the first turn is called Player 1
    - m < n
    - Both m and n are natural number
    - The player must remove at least one pebble
    - The player may remove up to every pebble from one pile
    - The player may not remove pebbles from both piles during the same turn
    - The player may choose a different pile in a different turn
    """
    remove_num = random.randint(a: 1, min(n, m))
    remove_pile = random.choice([n, m])
    if remove_pile == n:
        remove_pile -= remove_num
        return [remove_pile, m]
    else:
        remove_pile -= remove_num
        return [n, remove_pile]
```

```python
def win(n: int, m: int) -> str:
    """
    Impletment the strategy as a recursive function that takes the size of the two piles

    :param n: the number of pebbles in a pile, called pile 1
    :param m: the number of pebbles in another pile, called pile 2
    :return: the player who win the game

    Precondition:
    - The first player who takes the first turn is called Player 1
    - m < n
    - Both m and n are natural number
    - The player must remove at least one pebble
    - The player may remove up to every pebble from one pile
    - The player may not remove pebbles from both piles during the same turn
    - The player may choose a different pile in a different turn
    """
    # Use the strategy developed for player 1 get the return list as lst1
    lst1 = p1_win_strategy(n, m)
    # Check whether the number of pebbles in both piles is 0, if it's not, then continue the game
    if lst1[0] != 0 or lst1[1] != 0:
        # Use the random function developed for player 2 to continue the game, ge tthe return list as lst2
        # The input of the function will use the updated pebbles in both piles after implementing the strategy
        # developed for player1, which is the list lst1
        lst2 = p2_move_strategy(lst1[0], lst1[1])
        # Check whether the number of pebbles in both piles is 0, if it's not, then continue the game
        if lst2[0] != 0 or lst2[1] != 0:
            # Call the functino recursively and once both piles' pebbles become 0, the function will return the string
            # which player wins the game (In our strategy, it's player 1)
            a = win(lst2[0], lst2[1])
            return a
        # After checking, the number of pebbles in both piles is 0 after player 2 implementing, so player 2 win.
        else:
            return "Player 2 win!"
    # After checking, the number of pebbles in both piles is 0 after player 1 implementing, so player 1 win.
    else:
        return "Player 1 win!"
```