Problem 1. Find a formula for the Sprague-Grundy value of a pile of size \( n \) in the following subtraction games:
(a) Subtraction set is all multiples of 3.
(b) Subtraction set is \( \{2,3,7\} \).

Solution: (a) We show by induction that \( g(n) = \lfloor n/3 \rfloor \), where \( \lfloor x \rfloor \) is the integer part of \( x \).

For \( x = 0,1,2 \) there are no followers, so indeed \( g(x) = 0 \). For larger \( x \) the followers are \( \{x - 3, x - 6, \ldots\} \). If \( x = 3n \), then the followers are \( \{0, 3, 6, \ldots, 3n - 3\} \). By the induction hypothesis, these have \( g \) values \( \{0, 1, \ldots, n-1\} \) so the mex is \( n \). If \( x = 3n + 1 \), then the followers are \( \{1, 4, \ldots, 3n - 2\} \). If \( x = 3n + 2 \), then the followers are \( \{2, 5, \ldots, 3n - 1\} \). In these cases the \( g \)-values of the followers are also \( \{0, 1, \ldots, n-1\} \) so the mex is \( n \).

(b) Computing the first few cases shows that \( g(n) \) is periodic, with repeating values \( \{0, 0, 1, 1, 2\} \), so

\[
g(x) = \begin{cases} 
0 & (x \mod 5) \in \{0, 1\}, \\
1 & (x \mod 5) \in \{2, 3\}, \\
2 & (x \mod 5) = 4.
\end{cases}
\]

We check this by hand for \( x < 7 \). For larger \( x \), the followers are \( \{x - 2, x - 3, x - 7\} \), which are equivalent to \( x - 2 \) and \( x - 3 \) modulo 5. Checking the 5 possibilities for \( x \mod 5 \) completes the proof. For example, if \( x \equiv 1 \) (mod 5), then the followers are 3 and 4 modulo 5, and so have \( g \)-values 1, 2 and the mex is 0, as required.

Problem 2. Players play the following game with red, blue and green chips. A move is to take either any number of red chips, or at most 4 blue chips, or at most 6 green chips. Find the Sprague-Grundy value of the position \((R, B, G) = (25, 37, 41)\), and all winning moves from that position.

Solution: This is a sum of three games. The red game is nim, with \( g(R) = R \). The blue game is the subtraction game with at most 4 chips, so \( g(B) = (B \mod 5) \). The green game is the subtraction game with at most 6 chips, so \( g(G) = (G \mod 7) \). (The case of \( S = \{1, 2, \ldots, k\} \) was discussed in class.) Therefore, we have \( g(R, G, B) = R \oplus (B \mod 5) \oplus (B \mod 7) \).

For \((R, B, G) = (25, 37, 41)\) we have \( g(25, 37, 41) = 25 \oplus 2 \oplus 6 = 29 \). The blue game has no position with \( g \)-value 2 \( \oplus \) 29. The green game has no position with \( g \)-value 6 \( \oplus \) 29. Therefore the only winning move is in the red game, to 25 \( \oplus \) 29 = 4, so remove all but 4 red chips.

Problem 3. There are two piles of chips. A valid move is to take any number of chips from one of the piles (as in NIM) or to take a single chip from both piles. Find a formula for the Sprague-Grundy value \( g(n, 1) \).
Solution: If one of the piles is empty, this is just NIM, so \( g(n, 0) = n \oplus 0 = n \). From \((n, 1)\) the possible moves are to \((m, 1)\) with \( m < n \), or to \((n, 0)\) or to \((n-1, 0)\). 

2 0 4 5 3

Problem 4. Consider the partisan subtraction game with subtraction sets \( S_A = \{1, 3, 4\} \), and \( S_B = \{1, 2\} \). Find the outcome (type, from \{N,P,A,B\}) of all positions. Justify your answer.

Solution: The type of \( n \) is \( P \) if \( n \equiv 0 \pmod{5} \), it is \( B \) if \( n \equiv 2 \pmod{5} \), and all other \( n \) are \( N \)-positions. The proof is by induction. Suppose the claim holds for all \( m < n \).

First consider the case \( n = 5k \). Here Bob can move to \( 5k - 1 \) or \( 5k - 2 \), both \( N \)-positions. Alice can move to \( \{5k - 1, 5k - 3, 5k - 4\} \), also all \( N \)-positions by the induction hypothesis, so this is a \( P \)-position. It follows that Bob can win if he moves first from \( n = 5k + 1 \) or \( n = 5k + 2 \).

From \( 5k + 1 \), Bob can win by moving to \( 5k \). Alice can move to \( \{5k + 1, 5k - 1, 5k - 2\} \), all \( N \)-positions. Therefore Bob wins from \( 5k + 2 \) no matter whose turn it is.

Problem 5: Arithmetic progressions. Consider the following game. The board is a one sided strip of squares, labeled \( \{0, 1, 2, 3, \ldots, N\} \) in order. Each player in their turn places a stone of their colour in an empty square. A player wins if five of their squares form an arithmetic progression, namely they control squares \( \{a, a + d, a + 2d, a + 3d, a + 4d\} \) for some \( a, d > 0 \). It is known that \( N \) is such that the game cannot end in a draw. Show that the first player has a winning strategy.

Hint: Use strategy stealing.

Note: van der Waarden’s theorem says that even on a large enough finite board it is impossible for this game to be a draw, but it is not known how large the board needs to be. Tim Gowers proved this happens if \( N \geq 2^{2^{2^{2^{14}}}} \). Improving this bound is a difficult open problem.

Solution: Assume for a contradiction that player 2 (Bob) has a winning strategy. Player 1 (Alice) makes an arbitrary first move, and afterwards uses Bob’s winning strategy, pretending that the first stone is not there.

The presence of the first stone only makes that move not available to Bob, which does not harm Alice’s response. If the strategy directs Alice to play where the extra stone is, she just plays again at an arbitrary location and pretends the new stone is not there afterwards instead of the first stone.

Bonus problems (optional)

Problem 6. For the game from problem 3:

(a) find a general formula for \( g(x, y) \).

(b) Prove your the formula.
Solution: The formula: If \( x, y \in \{0, 1, 2\} \) then \( g(x, y) = (x + y) \mod 3 \). For larger values it turns out that for any \( x, y \leq 3 \) and any \( a, b \in \mathbb{N} \):

\[
g(3a + x, 3b + y) = g(x, y) + 3(a \oplus b).
\]

It is possible to prove this by checking many cases. A shorter proof uses a mixed base 2-3 representation. Write \( x = a_0 + 3 \sum a_i 2^i \), and \( y = b_0 + 3 \sum b_i 2^i \), with \( a_0, b_0 \in \{0, 1, 2\} \) and the others in \{0, 1\}. Then the claim is that

\[
g(x, y) = (a_0 + b_0 \mod 3) + \sum (a_i \oplus b_i)2^i.
\]

Thus digits of \( x, y \) are added without carry. To prove this is the Sprague-Grundy value, we need to show two things:

- If \((x', y')\) is a follower of \((x, y)\) then \(g(x', y') \neq g(x, y)\).
- For every \( s < g(x, y) \) there is a follower with \( g(x', y') = s \).

The first is obvious: changing some digits changes the sum. The second is similar to the case of NIM. If \( s \) differs from \( g(x, y) \) in some digit except the least, then the in first place this occurs \( s \) has digit 0 and \( g(x, y) \) has a 1 (since \( s < g(x, y) \)). This means we can remove chips from one pile to get to value \( s \).

The remaining case is when all but the last digit of \( s \) agree with \( g(x, y) \). Let the last digits of \( s \) and \( g(x, y) \) be \( s_0, c_0 \).

- If \( s_0 = 0, c_0 = 1 \) then one of \( a_0, b_0 \) is non-zero and we take one chip from that pile.
- If \( s_0 = 1, c_0 = 2 \) then also one of \( a_0, b_0 \) is non-zero and we take one chip from that pile.
- If \( s_0 = 0, c_0 = 2 \) and one of \( a_0, b_0 \) is 2, we take two chips from that pile.
- The remaining case is \( s_0 = 0, c_0 = 2, a_0 = b_0 = 1 \). Here we take one chip from each pile.

Problem 7. Write a python program to compute the Sprague-Grundy of a chomp board. After executing your file in python, there should be a function \texttt{chomp(A)} which takes a tuple \( A \) and returns the value of that board. For example, the 5×5 board is \( A=(5, 5, 5, 5, 5) \) and \texttt{chomp(A)} should return 6. If 3 squares are missing from the top row, this is described as \( (5, 5, 5, 5, 2) \). The lengths of rows are given from the bottom up. Solutions must be submitted online, instruction to follow.

Solution: My solution:

```python
def moves(B):  # possible moves from position B, given as a tuple.
    L = len(B)
    for i in range(L):
        for j in range(B[i]):
            if i==j==0: continue
            yield tuple((B[k] if k<i else min(B[k],j)) for k in range(L))

def mex(A):  # mex of a list or generator
    S = set(A)
    for i in range(len(S)+1):
        if i not in S: return i

@memoize
def g(x):
    return mex(g(y) for y in moves(x))
```

@memoize
def g(x):
    return mex(g(y) for y in moves(x))
Here, `@memoize` instructs Python to remember previously computed values of the function instead of re-computing them. With this, the formula is used once for each possible board configuration. Without it, the program will essentially go over all possible games, and even $4 \times 4$ is slow (millions of possible games). Include the following code.

```python
def memoize(f):
    """ Memoization decorator for functions taking one or more arguments. ""
    class memodict(dict):
        def __init__(self, f):
            self.f = f
        def __call__(self, *args):
            return self[args]
        def __missing__(self, key):
            ret = self[key] = self.f(*key)
            return ret
    return memodict(f)
```