Problem 1. Find the Nim-sum of all numbers from 1 to $2^n - 1$, where $n > 1$ is a natural number. More precisely, compute

$$1 \oplus 2 \oplus \cdots \oplus 2^n - 1.$$ 

Hint: Show that there are exactly $2^{n-1}$ numbers with 1 as the $k$th digit of their binary expansion for each $k = 1, 2, \ldots, n$.

**Solution:** Each sequence of digits $(a_0, \ldots, a_n)$ with $a_i \in \{0, 1\}$ corresponds to a number $a = \sum a_i 2^i$, and this gives all numbers from 0 to $2^n - 1$. If we want numbers where $a_k = 1$, there are exactly $2^{n-1}$ possibilities for the other digits. Therefore the nim-sum $0 \oplus 1 \oplus \cdots \oplus (2^n - 1)$ has $2^{n-1}$ ones in each position. Since $2^{n-1}$ is even, the required sum is 0.

**Alternative solution.** We can add pairs in the sum. Since $2^n \oplus (2n + 1) = 1$, we get

$$S = 1 \oplus (2 \oplus 3) \oplus (4 \oplus 5) \oplus \cdots = 1 \oplus 1 \oplus 1 \oplus \cdots = 0,$$

Since there are $2^{n-1}$ 1’s.

Problem 2. The columns of an $8 \times 8$ chess board are denoted by letters a–h, and the rows by numbers 1–8. A game is played with two rooks on a chess board, with the following rules: A valid move is to move one of the rooks either to the left within its row, or down in its column, but not to the right or up. Unlike in chess, the rooks can occupy the same square, and one can also jump over the other. For example, if the rooks are in c4 and c7, it is valid to move the c7 rook to c1.

Which player wins this game if initially the rooks are at d8 and g6? If it is the first player, what are the winning moves?

Hint: This is NIM in disguise.

**Solution:** Let a rook at row $i$ and in the $j$th column correspond to a pair of piles of sizes $i - 1$ and $j - 1$. For example d7 is piles of sizes (3, 6). A move in this game is to reduce the size of one of the piles. Since rooks do not interact, this is exactly NIM.

The given position \{d8, g6\} gives piles of sizes \{3, 7, 6, 5\}. The nim-sum of these is 7, so this is an N-position. The winning move is 7 $\rightarrow$ 0, or 6 $\rightarrow$ 1 or 5 $\rightarrow$ 2. These correspond to rook moves d7$\rightarrow$ d0, or g6$\rightarrow$ b6, or g6$\rightarrow$ g3.

Problem 3. 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 5.

(b) Subtraction set is \{2, 3, 7\}. 

Solution: (a) We show by induction that \( g(n) = \lfloor n/5 \rfloor \), where \( \lfloor x \rfloor \) is the integer part of \( x \). For \( x < 5 \) there are no followers, so indeed \( g(x) = 0 \). For larger \( x \) the followers are \( \{x-5, x-10, \ldots \} \). If \( x = 5n \), then the followers are \( \{0, 5, 10, \ldots , 5n-5\} \). By the induction hypothesis, these have \( g \) values \( \{0, 1, \ldots , n-1\} \) so the mex is \( n \). Similarly, if \( x = 5n + i \) for some \( i < 5 \), then the followers are \( \{i, i+5, \ldots , i+5n-5\} \). In all 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 \( x \equiv 1 \pmod{5} \), then the followers are 3 and 4 modulo 5, and so have \( g \)-values 1, 2 and the mex is 0, as required.

Problem 4. 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 6 blue chips, or at most 7 green chips. Find the Sprague-Grundy value of the position \((R, B, G) = (52, 73, 65)\), 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 \( g(B) = (B \mod 7) \). The green game is the subtraction game with at most 6 chips, so \( g(G) = (G \mod 8) \). (The case of \( S = \{1, 2, \ldots , k\} \) was discussed in class.) Therefore, we have \( g(R, G, B) = R \oplus (B \mod 7) \oplus (B \mod 8) \).

Therefore,
\[
g(52, 73, 65) = 52 \oplus (73 \mod 7) \oplus (65 \mod 8) = 52 \oplus 3 \oplus 1 = 54.
\]
The only winning move is to take 50 red chips, leaving 2 (since \( 2 \oplus 3 \oplus 1 = 0 \)). There is no winning move in the blue or green games.

Problem 5. 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 \((n-1, 0)\). We can find the first few values of \( g(n, 1) \) are \( 1, 2, 0, 4, 5, 3, 7, 8, 6, \ldots \) The general pattern is
\[
g(n, 1) = \begin{cases} 
n + 1 & n \equiv 0 \text{ or } n \equiv 1 \pmod{3}, \\
n - 2 & n \equiv 2 \pmod{3}.
\end{cases}
\]
To prove this, check the 3 cases. For example, if \( n = 3k \) than all \( i < 3k \) are values of \( g(m, 1) \) for \( m < n \), and \( g(3k, 0) = 3k \), so the mex is \( 3k + 1 \). The two other cases are similar.

**Bonus problem (optional)**

**Problem 6.** For the game from problem 5:
(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 \( x = a_0 + 3 \sum a_i2^i \) and \( y = b_0 + 3 \sum b_i2^i \), 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 \(\text{chomp}(A)\) which takes a tuple \(A\) and returns the value of that board. For example, the \(5 \times 5\) board is \(A=(5,5,5,5,5)\) and \(\text{chomp}(A)\) 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 to this must be submitted on canvas.
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))

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 × 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)