ICS3U Case Study: Nim

The game of **nim** has many variants, but usually involves two players taking turns removing/adding a number of objects from/to an existing amount, until a certain number is reached. When this number is reached, the first player either wins or loses, depending on the rules.

Take, for instance, a simplified version of Nim that begins with 21 objects in a **heap**. Players might be allowed to remove one, two or three objects from the heap per turn (no more, and no less). The player who takes the last object loses. A game in which the last player to remove an object is the loser is often referred to as a *misere* game. A typical game might go something like this.

```
Heap = 21    Player 1 takes 3 objects
Heap = 18    Player 2 takes 1 object
Heap = 17    Player 1 takes 3 objects
Heap = 14    Player 2 takes 2 objects
Heap = 12    Player 1 takes 2 objects
Heap = 10    Player 2 takes 3 objects
Heap = 7     Player 1 takes 1 object
Heap = 6     Player 2 takes 2 objects
Heap = 4     Player 1 takes 3 objects
Heap = 1     Player 2 loses
```

To create a program that will allow two people to play Nim, we can use the skills we have developed in the first half of this course. To begin with, we need a game loop that will run as long as there are objects in the heap. Inside of the loop, we must check that each move is valid – that is, a player only takes 1-3 objects from the heap. If this is true, we remove the objects from the heap, and switch players. Once there are no more objects, the player who took the last one loses. Below is code that does this.

```
# Plays a variant of Nim, with one heap of 21 objects.
# The player to remove the last object loses.
heap = 21
player = 1
while heap > 0:
   # display current player and heap count
   print("Player", player, ": The heap contains", heap, "objects.")
   # player can only remove 1, 2 or 3 objects from heap
   n = 0
   while n < 1 or n > 3:
      n = int(input("Remove how many objects? "))
   # remove objects from the heap
   heap -= n
   # toggle player
   if player == 1:
      player = 2
   else:
      player = 1
# display winner
print("Player", player, "wins!")
```

At the beginning of the program, there are 21 objects in the heap and it is player 1's turn. As long as `heap > 0`, there are objects in the heap, and the game starts/continues. The nested `while` loop validates a player's input: if the player tries to remove any number of objects outside of the rage 1-3, then one of the two conditions `n < 1` or `n > 3` is `True` and the loop will repeat the prompt. If the player enters a valid number of objects, the program removes them from the heap and switches from player 1 to player 2, or *vice versa*. This continues until there are no objects in the heap, at which point the outer loop finishes and the winner is announced. Try playing a few times, using different values each time.

Depending on the values you chose, you might have discovered one small change that should be made. In the code above, it is possible for a player to remove more objects than are in the heap at the end of the game. Here is an example that illustrates this.

```
Player 1 : The heap contains 21 objects.
Remove how many objects? 3
Player 2 : The heap contains 18 objects.
Remove how many objects? 3
Player 1 : The heap contains 15 objects.
Remove how many objects? 3
Player 2 : The heap contains 12 objects.
Remove how many objects? 3
Player 1 : The heap contains 9 objects.
Remove how many objects? 3
Player 2 : The heap contains 6 objects.
Remove how many objects? 3
Player 1 : The heap contains 3 objects.
Remove how many objects? 2
Player 2 : The heap contains 1 objects.
Remove how many objects? 3
Player 1 wins!
```

While this is a minor issue (the player still loses, after all), it is easy to fix. In the validation loop, we add the condition that the number of objects taken cannot exceed the number that are in the heap. The finalized code would look something like this. Note the third condition in the inner `while` loop.

```
# Plays a variant of Nim, with one heap of 21 objects.
# The player to remove the last object loses.
heap = 21
player = 1
while heap > 0:
    # display current player and heap count
    print("Player", player, ": The heap contains", heap, "objects.")
    # player can only remove 1, 2 or 3 objects from heap
    n = 0
    while n < 1 or n > 3 or (heap - n < 0):
        n = int(input("Remove how many objects? "))
    # remove objects from the heap
    heap -= n
    # toggle player
    if player == 1:
        player = 2
    else:
        player = 1
# display winner
print("Player", player, "wins!")
```

If there is one object in the heap, and the player tries to remove three objects as in the example above, then `heap - n < 0` is `True`, since 1 - 3 = -2 < 0. This ensures that the inner loop prompts the user to enter a valid value.

Write programs that play each variant of Nim, all of which are *misère* versions. Try to keep your code as simple as possible.

- **Twenty-One**: Beginning with a value of one, two players alternate saying a number that is 1-3 greater than the previously mentioned number. The player who is forced to say twenty-one loses. For example, a game may be 1, 4, 6, 9, 10, 13, 16, 17, 19, 20, 21.
- **Multi-Heap Nim**: Beginning with three heaps, each containing any (possibly different) number of objects, players may remove *any* number of objects from a single heap.
- **Wythoff's Game**: Beginning with two heaps, each containing any (possibly different) number of objects, players may remove either any number of objects from a single heap, *or* the same number of objects from *both* heaps.