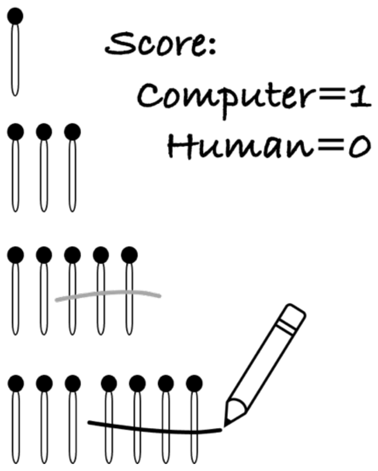
Laboratory 3 – Game of Nim

The game of Nim is an ancient Chinese game originated (捡石子 *jiǎn-shízi*) originally played with stones.

The game starts with a certain number of heaps with a different number of objects (stones, coins, etc.). At each turn, players remove any number of objects from a heap. The player that removes the last object wins (or loses depending on the variant).

Nim is a *winning strategy* game. This means that who plays first is always guaranteed to win if certain rules are followed.

Can you beat the computer?



Description

First, the program prompts the user for the number of heaps (between 2 and 5) and initialises an array of integers using as sequence of odd numbers (1, 3, 5…) to hold the number of coins in each heap. Then, players alternate moving with the user playing first.

At the start of a round, the program shows the remaining coins by printing a row of "o" for each heap. Then, the program prompts the user for the heap and coins to remove and updates the array.

Next, the program determines its move (heap/coins) by using the *XOR sum* of the heap sizes. This is a sum that uses the *bitwise XOR* "^" instead of the "+". A XOR sum of zero indicates a losing position. So, the program tries to force a zero XOR sum for the opponent with its move.

At the end of the cycle the array and *total* are updated according to the move. When a player takes last coin, the player wins and the program terminates printing a final message. The printout on the right shows an example of output.

```
How many heaps [2-5]? 4

------------------ round 1
heap[0]: o
heap[1]: ooo
heap[2]: ooooo
heap[3]: ooooooo
Your Turn: input heap, coins: 2 2
My Turn: heap, coins: 3 6

------------------ round 2
heap[0]: o
heap[1]: ooo
heap[2]: ooo
heap[3]: o
Your Turn: input heap, coins: 1 2
My Turn: heap, coins: 2 2

------------------ round 3
heap[0]: o
heap[1]: o
heap[2]: o
heap[3]: o
Your Turn: input heap, coins: 0 1
My Turn: heap, coins: 1 1

------------------ round 4
heap[0]:
heap[1]:
heap[2]: o
heap[3]: o
Your Turn: input heap, coins: 2 1
My Turn: heap, coins: 3 1

Sorry, you lose
```

Write the program through the following steps:

1. Define the macro constants (using #define) MIN_HEAPS for the minimum number of heaps, and MAX_HEAPS for the maximum number of heaps.
2. Define an array with MAX_HEAPS integers to hold the number of coins in each heap.

3. Define and initialise any other required variables. These include the total number of coins (*int total*), the number heaps (*int heaps*), and the round counter (*int round*).
4. Input the number of heaps from keyboard. The input should be checked, and the user should be prompted again if the input is not valid.
5. Initialise the array with a for-loop using the odd number pattern (heap[0]=1, heap[1]=3, etc.) and initialise *total*. Remember to declare the indexes to scan the array in of the for-loop too.
6. Loop until coins are available (*total>0*). Alternate between user and computer (a variable to keep track whose turn it is may be needed). At the end of the loop update the array and the coins.
7. On user turn:
   a. Print the round number and the coins in each heap coins.
   b. Input heap/coins from keyboard. The input should be checked, and user prompted again if the input is not valid.
8. On computer turn, the move (heap/coins) uses the following steps:
   a. Calculate *nimsum* as the "XOR sum" of all the heap sizes:

   ```
   /* Nim sum*/
   nimsum = 0;
   for(i=0; i<heaps; i++)
       nimsum = nimsum ^ heap[i];
   ```

   b. If *nimsum* is positive, determine the heap such that *nimsum XOR heap* is strictly less than the heap size, and take as many coins on that heap as needed to leave the heap with *nimsum XOR heap* coins.
   c. If *nimisum* is zero (a losing position for the machine), remove one coin from the first available heap.

9. Print a final message when the game terminates.