Code Jam 2010 - Qualification Round

Analysis: Snapper Chain

The difficult part was understanding how a single Snapper works. Each Snapper can be in one of two states -- On or Off. Also, each snapper can either be powered or unpowered. The first Snapper is always powered because it is plugged into the power socket in the wall. The i'th Snapper is powered if and only if the (i-1)'th Snapper is powered and On. Snapping your fingers changes the state of each powered snapper (from On to Off, or from Off to On).

With these rules in mind, let's represent the On/Off state of the Snappers by a sequence of bits, 1 meaning On. If we list the bits right-to-left, we get a binary integer. Initially, the integer has value 0. Similarly, we can write down the binary integer for the powered/unpowered state of each Snapper. Initially, this integer is 1 because only the rightmost Snapper is powered.

Snapping your fingers is equivalent to doing an XOR of the On/Off integer and the powered/unpowered integer and putting the result into the On/Off integer. After that, we update the powered/unpowered bits according to the rule above,

For example, let's say that the On/Off number is now 10100011111. This means that the powered/unpowered number is 00000111111. When we XOR these two numbers, we get the new value of the On/Off number: 10100100000.

The interesting thing is that these two updates are equivalent to a simple increment of the On/Off integer! Snapping your fingers adds 1 to the On/Off integer, and we do not even need to care about the powered/unpowered integer.

The solution to the problem is then very simple. The answer is "ON" if and only if the rightmost **N** bits of **K** are 1.