Problem 9.1 **Pewlett Hackered Printer Problem** (page 1 of 2)

Overview: Interactively find the missing number in an array by querying bits.

Description: Pewlett Hackered (PH), the famous printer company, just released the brand-new PH-x series! These printers can print a bedazzling variety of colors. The PH-x256, for example, has 256 cartridges of colors, numbered 0 through 255. In order to maintain their vibrant PH-level of printing and keep shady companies from producing imitations, the cartridges are arranged in a random order inside the printer.

You recently purchased a PH-x*n* printer containing *n* cartridges, but it seems that some PH employees fell asleep on the job, so exactly one random cartridge of every new PH-x printer is missing from each model (including your PH-x*n* printer). The super-sensitive security mechanisms prevent anyone from opening the printer after it has been manufactured, so you can't look inside to find the number of the missing cartridge.

However, it turns out that it is actually possible to find out information about the cartridges by querying the printer for a bit of information at a time. This can be done up to 2*n* times on a PH-x*n* until the patented PH-Copy-Safe™ self-destruct mechanism activates. Can you find the number of the missing cartridge and salvage your printer?

Each PH-x*n* printer is designed to contain exactly *n* cartridges in random order, where *n* is a power of 2. Inside a broken PH-x*n* printer, exactly one cartridge is missing, so there are only *n*-1 cartridges present. The cartridges are stored in a secret array with *n*-1 distinct integers from 0 through *n*-1. Your program must find out which cartridge number is missing by querying the *j*-th binary digit of the *i*-th array element. We will return the *j*-th digit indexing from the least significant digit (i.e. counting digits from the right) at 0. You can make up to 2n queries, after which you must output the number of the missing cartridge.

For example, consider the array {1, 3, 0, 7, 2, 5, 4} for *n*=8 and *m*=6. In binary, this array holds {001, 011, 000, 111, 010, 101, 100}. Here are some sample queries (output) and responses (input):

Output	Secret array (behind the scenes)	Input
`0 0`	{00**1**, 011, 000, 111, 010, 101, 100}	`1`
`4 0`	{001, 011, 000, 111, 01**0**, 101, 100}	`0`
`4 1`	{001, 011, 000, 111, 0**1**0, 101, 100}	`1`
`4 2`	{001, 011, 000, 111, **0**10, 101, 100}	`0`
`6 2`	{001, 011, 000, 111, 010, 101, **1**00}	`1`
`7 2`	{001, 011, 000, 111, 010, 101, 100}	Program terminated (Out of bounds)
`-1 0`	{001, 011, 000, 111, 010, 101, 100}	Program terminated
`3 8`	{001, 011, 000, 111, 010, 101, 100}	`0` (All numbers have infinite leading 0's)
`3 -1`	{001, 011, 000, 111, 010, 101, 100}	Program terminated

Problem 9.1 **Pewlett Hackered Printer Problem** (page 2 of 2)

Time Allocation: 1 second

Input/Output: This is an interactive problem. This means that your program will receive input from the grading environment based on the output your program produces. All input and output will be done through the console.

Rules of interaction:

1. Your program should begin by reading in a single integer *n*.
2. Then, for each of up to 2*n* queries, your program should output a single pair of integers *i j*, separated by a single space.
3. You MUST output a new line character and flush the output stream after each output!
4. Each query will result in an integer response *k*, representing the *j*-th binary digit of the *i*-th element of the secret array.
5. At any time in place of the 2*n* queries, or immediately following the 2*n*-th query, your program may output a single integer *m* representing the missing cartridge. This should be the final output from your program.

Assumptions and Expectations:
n will be an integral power of 2, from 2 to 4096, inclusive.
i should be an integer between 0 and *n*-2, inclusive.
j should be an integer greater than or equal to 0.
The response *k* will be either `0` or `1`.
If any output is invalid, your program will be deemed incorrect.
Only one attempt at *m* is allowed.
No more than 2*n* queries will be accepted. After 2*n* queries, your program must output a single integer *m*. Otherwise, your program will be terminated and deemed incorrect.

Sample Run:

```
 Input: 4
Output: 0 0
 Input: 1
Output: 0 1
 Input: 1
Output: 1 0
 Input: 0
Output: 1 1
 Input: 0
Output: 2 0
 Input: 0
Output: 2 1
 Input: 1
Output: 1
```

(The cartridge array was {3, 0, 2}, or {11, 00, 10} in binary.)