HKOI 2014/15 Junior Inverse Problem

Problem setter Alex Poon

Presenter: Steven Lau

Inverse Problem

- Input: 2 integers **N M**
  - $1 \leq$ **N** ≤ 500000
  - $-2^{31} \leq$ **M** $\leq 2^{31}-1$
- Output: **N** distinct integers
  - $-2^{31} \leq$ **$a_1,a_2,a_3,\ldots,a_N$** $\leq 2^{31}-1$
  - **$a_1+a_2+a_3+\ldots+a_N$ = M**
  - If there are multiple solutions, output any one of them

Inverse Problem

- Input: 2 integers **N M**
  - $1 \leq$ **N** ≤ 500000
  - $-2^{31} \leq$ **M** $\leq 2^{31}-1$
- Sample

Input	Output
3 6	1 2 3
4 8	3 -1 4 2

Think functional?

- f(N, M) = [a_1,a_2,a_3,...,a_N] where
  - 1 ≤ N ≤ 500000
  - -2^{31} ≤ M ≤ 2^{31}-1
  - -2^{31} ≤ a_1,a_2,a_3,...,a_N ≤ 2^{31}-1
  - a_1+a_2+a_3+...+a_N = M
- Find f.

Adhoc

- `f(1, M) = [M]`

Input	Output
1 -2147483648	-2147483648
1 -123	-123
1 -1	-1
1 0	0
1 456	456
1 2147483647	2147483647

Adhoc

- `f(2, M) = [M, 0]`

Input	Output
2 -2147483648	-2147483648 0
2 -123	-123 0
2 -1	-1 0
2 0	1 -1
2 456	456 0
2 2147483647	2147483647 0

Adhoc

- `f(3, M) = [M,1,-1]`

Input	Output
3 -2147483648	-2147483648 1 -1
3 -123	-123 1 -1
3 -1	-1 2 -2
3 1	1 2 -2
3 456	456 1 -1
3 2147483647	2147483647 1 -1

Subtask 1: 1 ≤ N ≤ 3

30 points in pocket!

continue to observe

When N = 1,3,5,7,9,...

- f(1, M) = [M]
- f(3, M) = [M,1,-1]
- f(5, M) = [M,1,-1,2,-2]
- f(7, M) = [M,1,-1,2,-2,3,-3]
- f(9, M) = [M,1,-1,2,-2,3,-3,4,-4]
- ...
- Yeah, full score, early leave

Wait!

- The result may not be distinct!
- For example,

  f(7, 2) will be [2,1,-1,2,-2,3,-3]
- Let's make it   [2,1,-1,3,-3,4,-4]

When N = 2,4,6,8,...

- f(2, M) = [M,0]
- f(4, M) = [M,0,1,-1]
- f(6, M) = [M,0,1,-1,2,-2]
- f(8, M) = [M,0,1,-1,2,-2,3,-3]
- …
- Not yet full score

When N = 2,4,6,8,...

- f(8, -1) = [-1,0,1,-1,2,-2,3,-3]
- make it to [-1,0,2,-2,3,-3,4,-4]
- full score, go on to next problem

You wish >:]

- f(8, 0) = [0,0,1,-1,2,-2,3,-3]
- Make it   [1,-1,2,-2,3,-3,4,-4]

Thank you

```haskell
--Solution in Haskell for your interests
import Data.List


main = do
  [n, m] <- fmap (map read . words) getLine
  putStrLn . unwords . map show . take n $ f n m


zero = [1..] >>= (\x -> [x, -x]) --zero is [1,-1,2,-2,3,-3,...]


f n 0 | even n = zero

f n m | even n = 0 : m : (zero \\ [m, -m]) --remove [m, -m] from zero
      | odd  n =     m : (zero \\ [m, -m])
```