Cat’s Game

Objective

Give practice with binary search.

Story

After collecting your money from the kings and queens of chessland you begin to work your way back to the bureaucratic nightmare that is the capital of Gameland; got to get paid for that handle generator (hopefully it’s still working).

As you venture through the Forest of Wonders you notice your coin pouch is particularly lighter than usual. You look down only to notice that the once heavy leather bag caring your earnings has been replaced with a larger than average turnip. Laughing fills the air as a feline like apparition materializes on a tree above.

The cat introduces themselves as Patty Kerry, and they inform you that in order to retrieve your sack of gold you need to play a simple game. They are thinking of a number between 1 and n, and you need to guess the number in as few as guesses as possible. For every guess you make that is wrong Patty will eat some of your coins.

Luckily, Patty is willing to give you hints. For every wrong guess after your first guess, if your current guess is closer to the target they will say “No. Warmer.” If your current wrong guess is farther from the target they will say “No. Colder.” Lastly, if your wrong guess is the same distance Patty will say “No. No change.”

Problem

Given a range of numbers to guess ([1,***n***]), guess the correct integer in the range in less than 1+2*cieling(log$_2$(***n***)) tries.

Input Output Interaction

You should begin by reading a line containing 1 integer, ***n*** (1 ≤ ***n*** ≤ 1,000,000), representing the maximum positive integer that Patty can choose.

You will then perform a series of guesses. Each guess will requiring your program to output an integer in the range of [-1000000000, 1000000000]. Values outside that range will be treated as invalid. For a valid guess a single line will be output. If the guess is correct the line will begin with “Yes!!!”. If the guess is incorrect the line will begin with “No.”. For every guess after there is extra output. If the value is closer than the previous to the target, the word “Warmer.” will be printed separated from the first word by a space. If the value is further than the previous to the target, the word “Colder.” will be printed separated from the first word by a space. If the value is neither closer nor further than the previous to the target, the words “No change.” will be printed separated from the first word by a space. If your program takes too many guesses after the last “No.” line the phrase “Game Over.” will be outputted.

IMPORTANT: After every printf(...); in your program please use the line of code `fflush(stdout);` to tell my judger to read in the next line.

Sample Interaction 1

Input (The text your program reads in)	Output (your program's prints)
10	
	5
No.	
	10
No. Colder.	
	3
No. Colder.	
	7
Yes!!!	

Sample Interaction 2

Input	Output
20	
	3
No.	
	15
No. Warmer	
	20
No. Colder.	
	10
No. Warmer	
	16
No. No change.	
	13
Yes!!!	

Explanation

Interaction 1

The answer is 7 with a maximum value of 10. Your program only reads that 10 is the maximum value.

You can try any value you want, but the sample tries the value 5 first. That is not correct.

The next attempt the sample tries is 10. The value 5 was closer to 7 than what 10 is, because the difference between 5 and 7 is 2, but the difference between 10 and 7 is 3. Thus the output is “No. Colder.”

The next attempt the sample tries is 3. The value 3 is further from 7 than what 10 was, because the difference between 7 and 3 is 4, where the difference between 7 and 10 is 3. Thus the output is yet again “No. Colder.”

7 is the only value that fits that pattern, so 7 is outputted and the next line of input given to the program is “Yes!!!”. After “Yes!!!” is read in, the program needs to terminate (return 0).

Interaction 2

The answer is 13 with a maximum value of 20. The first guess which is wrong was 3.

The second guess which is wrong but closer (“Warmer”) is 15. The value is warmer because the difference between 3 and 13 = 10, but the difference between 15 and 13 = 2.

The third guess which is wrong and further (“Colder”) is 20. The value is colder, because the difference between 15 and 13 = 2 and the difference between 20 and 13 = 7.

The fourth guess which is also wrong (“Warmer”) was 10. The value is warmer because the difference between 10 and 13 = 3, but the difference between 20 and 13 = 7.

The fifth guess which is also wrong (“No change.”) was 16. The value has no change because the difference is identical between the two guesses (3).

The last guess is 13, because we should know that the midpoint would be the answer.

Hints

Reading Input: Keep track of the range of possible values (use a low and high value). When you get some information (warmer/colder/no change) you should be able reduce the range of possible values, assuming the program asks meaningful values.

Out of Bounds: Asking values that are negative or greater than the max can sometimes provide better information than asking values that are in the bounds of the possible values. As long as you don't print values outside the bounds of -1,000,000,000 to 1,000,000,000 I will give a response.

Binary Searching the Range: Try to ask pairs of values that will help split the range in half. Be careful that you don't ask the same value twice in a row!

Grading Criteria

- Read/Write from/to **standard** input/output (e.g. scanf/printf and no FILE *)
    - 10 points
- Good comments, whitespace, and variable names
    - 15 points
- Read in string responses from the user
    - 5 points
- Find a method to roughly split the space in half
    - 10 points
- Use values to keep track of the range of possible values
    - 10 points
- Programs will be tested on 10 cases
    - 5 points each

No points will be awarded to programs that do not compile using "gcc -std=gnu11 -lm".

Sometime a requested technique will be given, and solutions without the requested technique will have their maximum points total reduced. For this problem use a binary search to reduce the search space. ***Without this programs will earn at most 50 points!***

Any case that causes a program to return a non-zero return code will be treated as wrong. Additionally, any case that takes longer than the maximum allowed time (the max of {5 times my solutions time, 10 seconds}) will also be treated as wrong. Cases that do not terminate but get a Yes!!! verdict will still be treated as wrong.

No partial credit will be awarded for an incorrect case.