7087 The Binary Search Efficiency Doubter

The binary search is a classic algorithm in computer science. For this problem we will use the following pseudocode to define our binary search:

```
Perform a binary search on the array a with values in a[0] through a[n-1] to find,
if it exists, the value x.
The values in the array a may be assumed to be in strictly increasing order.

    Low = 0
    High = n - 1
    While Low <= High
        Mid = (Low + High) / 2 [We assume integer division truncates.]
        If a[Mid] = x then return FOUND
        If a[Mid] < x then Low = Mid + 1
        If a[Mid] > x then High = Mid - 1
    If we fall out of the while loop, return NOT_FOUND
```

Professors teach that this is an efficient algorithm with a worst case number of times through the loop of roughly log base 2 of n and an average case that is slightly better than that. A student who is not convinced decides to build lists of various sizes and search for every number in the list and keep track of how many times the loop is executed. In the following example, the number of times the loop is executed to find each value is indicated below the corresponding value.

the list	12	16	23	34	42	57	65
loop count	3	2	3	1	3	2	3

So, for this list, the total loop count is 17.

It should be clear that any list of length 7 will have a total loop count of 17 under the assumptions that the list is sorted and all the values are different. That is, the length of the list determines the total loop count.

The problem here is to determine the total loop count given the length of the list. You may assume that the answer for any test case in the input fits in a signed 64-bit integer.

Input

There may be multiple cases. Input for each case will consist of a single positive integer, n, which gives the length of the list. You may assume that $2 < n < 10,000,000$ and that there are no more than 100 cases. Cases are delimited by arbitrary white space. Process until an end-of-file is detected.

Output

For each input case, print the total loop count to find all the values in a list of size n. Follow this format exactly: 'Case', one space, the case number, a colon and one space, and the answer for that case with no trailing spaces.

Sample Input

```
3
7
321
124
```

Sample Output

```
Case 1: 5
Case 2: 17
Case 3: 2387
Case 4: 748
```