Counting Bits

Given an integer, *n*, determine the following:

1. How many 1-bits are in its binary representation?

2. The number *n*'s binary representation has *k* significant bits indexed from 1 to *k*. What are the respective positions of each 1-bit, in ascending order?

In the binary representation of 37, there are three 1-bits located at the respective 1st, 4th, and 6th positions.

Note: The leftmost 1 bit is always position 1. Preceding zeros are not considered in determining the position.

Function Description

Complete the function *getOneBits* in the editor below. The function must return a *results* array with the number of 1's stored at *results[0]* followed by the positions of all 1's in its binary representation in ascending order.

getOneBits has the following parameter(s):

n: an integer

Constraints

- $1 < n < 10^9$

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The single input is an integer, *n*.

Sample Case 0

Sample Input

```
161
```

Sample Output

```
3

1

3

8
```

Explanation

The integer $n = (161)_{10}$ converts to $(10100001)_2$:

In the binary representation of 161, there are 3 1-bits located at the 1st, 3rd, and 8th positions.

Because there are three 1-bits, the return array is $3 + 1 = 4$ units in length. Store the 1's count, 3, at index 0. Then store the locations of the 1-bits in order, low to high. Return the array [3, 1, 3, 8] as the answer.

```
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.function.*;
import java.util.regex.*;
import java.util.stream.*;
import static java.util.stream.Collectors.joining;
import static java.util.stream.Collectors.toList;
class Result {

    /*
     * Complete the 'getOneBits' function below.
     *
     * The function is expected to return an INTEGER_ARRAY.
    * The function accepts INTEGER n as parameter.
    */

   public static List<Integer> getOneBits(int n) {
   ******************************************************************
*******************************************************************
     }

   }
   public class Solution {
       public static void main(String[] args) throws IOException {
           BufferedReader bufferedReader = new
BufferedReader(new InputStreamReader(System.in));
           BufferedWriter bufferedWriter = new
BufferedWriter(new
FileWriter(System.getenv("OUTPUT_PATH")));

           int n =
Integer.parseInt(bufferedReader.readLine().trim());

           List<Integer> result = Result.getOneBits(n);

           bufferedWriter.write(
               result.stream()
                   .map(Object::toString)
                   .collect(joining("\n"))
               + "\n"
           );

           bufferedReader.close();
           bufferedWriter.close();
       }
   }

   ******************************************************************
*******************************************************************
```