Base Conversion

Write a program to convert a whole number specified in any base (2…16) to a whole number in any other base (2…16). *Digits* above 9 are represented by single capital letters: 10 by A, 11 by B, 12 by C, 13 by D, 14 by E and 15 by F.

Input

The input consists of three values:

- a positive integer indicating the base of the number
- a positive integer indicating the base we wish to convert to
- the actual number in the firs base that we wish to convert to the second base (this number will have letters representing any digits higher than 9; **it may have invalid *digits***)

Output

The output should consist of the original number followed by the string "base", followed by the original base number, followed by the string "=" followed by the converted number, followed by the string "base" followed by the new base.

If the original number is invalid, output the statement:

`originalValue` is an illegal base `originalBase` number

where `originalValue` is replace by the value to be converted and `originalBase` is replaced by the original base value.

Example 1

```
Input:
2 10 10101
Output:
10101 base 2 = 21 base 10
```

Example 2

```
Input:
5 3 126
Output:
126 is an illegal base 5 number
```

Example 3

```
Input:
15 11 A4C
Output:
A4C base 15 = 1821 base 11
```

Big- or Little- Endian

All computer memory is organized into 8-bit bytes. For integer values that require a type with more than 8-bits, such as the typical 2-byte, 4-byte, and 8-byte integral types available on most modern hardware, there is no universal agreement as to how to order those bytes and two incompatible storage schemes exist.

The first stores integers as groups of consecutive 8-bit bytes with the least significant byte occupying the lowest memory address within the group and the most significant byte occupying the highest memory address.

The second is just the reverse; the least significant byte is stored in the highest memory address within the group, and the most significant byte is stored in the lowest memory address.

Those schemes have been dubbed Little Endian and Big Endian, respectively.

We assume that signed integers are stored using "“"two’s complement" representation.

When binary integer data is shared between a Little Endian and Big Endian machine, a data conversion must be performed which involves reversing the bytes of the data. Once the bytes have been reversed the integer is then correctly interpreted by the hardware as the original value from the opposite-endian machine. Your task is to write a program that will read a list of integers and report the integers that the binary representations of the input integers would represent on an opposite-endian machine.

Input

A number `n` , `-2147483648 <= n <= 2147483647` .

Output

The number `n` followed by the phrase "converts to" followed by the other endian-value (single space between the three parts).

Example 1

```
Input:
123456789
Output:
123456789 converts to 365779719
```

Example 2

```
Input:
-123456789
Output:
-123456789 converts to -349002504
```

Example 3

```
Input:
1
Output:
1 converts to 16777216
```

Example 4

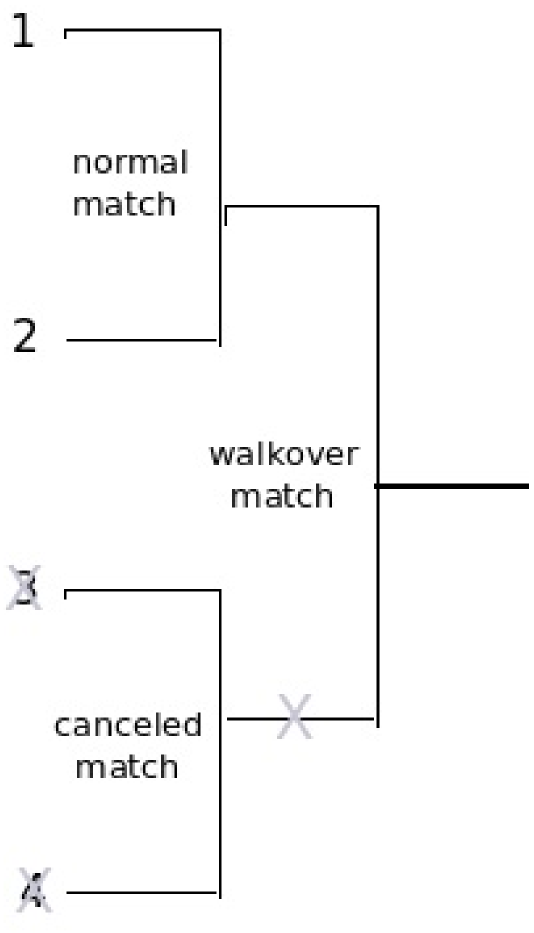
```
Input:
16777216
Output:
16777216 converts to 1
```

Chess Championship

In the World Chess Championship of 2035, 2^N contestants were supposed to participate. Because of sudden turn of the hurricane Alice, many flights were canceled last minute and many of the contestants could not arrive on time to participate in the championship.

With all the matches pre-arranged ahead of time, the organizer decide to keep the schedule as is and allow *walkover matches* in case when one of the players is missing.

- If both players are available, then there will be a normal match.
- If only one player is available, then this is a *walkover match* and the player automatically advances to the next level.
- If no player is available, then there is no match.



In the figure on the right, the players 3 and 4 could not arrive. Their match is canceled. Players 1 and 2, play their regular match and the winner advances to the next level match. But there is no second player there, so it is a *walkover match*.

Given the list of players who could not arrive on time, calculate the number of the *walkover matches* in the whole championship.

Input

Each test begins with two integers: `1 <= N <= 10` and `0 <= M <= 2^N`. `2^N` is the number of players scheduled for the championship. `M` is the number of players who could not arrive on time due to the canceled flights.
The next line contains `M` integers, denoting the players who have not arrived. The players are numbered `1` to `2^N`.

Output

The number of *walkover matches*.

Example 1

```
Input:
2 2
3 4
Output:
1
```

Example 2

```
Input:
3 5
1 2 3 4 5
Output:
2
```

Example 3

```
Input:
2 1
2
Output:
```

```
1
```

Negative base

Have you ever heard of base-2 (binary) numbers?

Well, of course you did!!! You are a computer scientist.

How about base-(-2) numbers? Yes, you read it right we are talking about a base that is negative!!!

An integer *n* writte in in base-(-2) is a sequence of bits (`b_i`), written right to left. Each of which is either `0` or `1` (no negative bits are allowed) and the following must hold:

```
n = b_0 + b_1 * (-2) + b+2 * (-2)^2 + b_3 * (-2)^3 + ...
```

Every integer in base-10 has a unique representation in base-(-2) and no negative signs are ever needed.

Your task is to write a program that given a number in base-10 finds its base-(-2) representation.

Input

The number in base-10, -1,000,000,000 <= N <= 1,000,000,000.

Output

The corresponding number in base-(-2) with no leading zeros.

Example 1

```
Input:
1
Output:
1
```

Example 2

```
Input:
7
Output:
11011
```

Example 3

```
Input:
-2
Output:
10
```

Example 4

```
Input:
0
Output:
0
```

Splitting Numbers

We define the operation of splitting a binary number `n` into two numbers `a(n)` and `b(n)` as follows. Let `0 <= i1 < i2 < ... < ik` be the indices of the bits (with the least significant bit having index 0) in `n` that are 1. Then the indices of the bits of `a(n)` that are 1 are `i1`, `i3`, `i5`, … and the indices of the bits of `b(n)` that are 1 are `i2`, `i4`, `i6`, …

For example, if `n` is `110110101` in binary then, again in binary, `a = 010010001` and `b= 100100100`.

Input

The input consists of a single integer `n` between `1` and `2^31 - 1` written in standard decimal (base 10) format on a single line.

Output

The output consists of a single line, containing the integers `a(n)` and `b(n)` separated by a single space. Both `a(n)` and `b(n)` should be written in decimal format.

Example 1

```
Input:
6
Output:
2 4
```

Example 2

```
Input:
7
Output:
5 2
```

Example 3

```
Input:
13
Output:
9 4
```