UTF-8 Validation

First, we need to know what a character is. A character is a printed or written symbol. In computing, a character is made from 1 to 4 bytes long. For 1 byte characters, the first bit is a 0 followed by its unicode code. For n-bytes characters, the first n-bits are all 1's, the n-1 bit is 0. Followed by n-1 bytes with most significant 2 bits being 10.

Char. number range (hexadecimal)	UTF-8 octet sequence (binary)
0000 0000–0000 007F	0xxxxxxx
0000 0080–0000 07FF	110xxxxx 10xxxxxx
0000 0800–0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx
0001 0000–0010 FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

So, given an array of integers representing the data, return true if it is a valid input.

The Input

The input of this program is only an array of integers. Only the least significant 8 bits are used to store data. Every integer represents 1 byte of data.

See examples below:

①
```
data = [197, 130, 1], which represents the octet sequence: 11000101 10000010 00000001.

Return true.
It is a valid utf-8 encoding for a 2-bytes character followed by a 1-byte character.
```

②
```
data = [235, 140, 4], which represented the octet sequence: 11101011 10001100 00000100.

Return false.
The first 3 bits are all one's and the 4th bit is 0 means it is a 3-bytes character.
The next byte is a continuation byte which starts with 10 and that's correct.
But the second continuation byte does not start with 10, so it is invalid.
```

An approach

For every token (integer), convert it to binary

If it is 8 bits, then proceed to check if the left most digit is 0. // Condition for UTF-8 Validity.

```
    If leftmost digit = 0
        return true
    else
        return false
else    // More than 8 bits ...
```

n-bits
1
n-1 bytes

① check that the first n-bits are all 1's
② check that the n+1th bit is 0
③ check that remaining n-1 bytes with the most significant 2 bits is 10.

} Conditions for n-bytes character

```
if ( 1 && 2 && 3 )
    return true
else
    return false
```

Complexity = O(n)