T9 Texting

Way back when, cell phones didn't have giant finger-friendly touchscreens. In fact, they often folded in half to fit in your pocket more easily. The input mechanism was a standard 12-key phone dialpad. On these keys are 3-4 letters, which represent the 26 letters of the English alphabet.

One of the first predictive text input systems was developed in the late 1990s and it worked like this:

```
| 1  |  | 2  |  | 3  |
|    |  |A B C|  |D E F|

| 4  |  | 5  |  | 6  |
|G H I|  |J K L|  |M N O|

| 7  |  | 8  |  | 9  |
|PQRS |  |T U V|  | WXYZ|

| *  |  | 0  |  | #  |
|    |  | _  |  |    |
```

If the user entered 4-7-3-2-8, that would resolve to "GREAT". However, if the user entered 4-6-6-3, that could resolve to:

- HOME
- GONE
- GOOD
- HOOD

Problem

The patents are about to expire, so I would like to rebuild T9 texting in order to corner the market. I will give you a dictionary and an input sequence; please turn this into valid words.

```
const stubbed_dictionary = {
  // This is just a fake dictionary for testing purposes.
  isWord: input => input.length % 2 === 0,
};

/**
 * Find all of the valid (according to {@code dict#isWord(string)}) words which
 * can be formed from the given input sequence.
 *
 * If you want to use a different input format, alternatives can be discussed.
 *
 * @param {string} input the sequence of buttons pressed
 * @param {{ isWord: (in: string) => boolean }} dict the dictionary implementation
 */
function findWords(input, dict) {
  // TODO: Find all of the words which can be created
}

console.log(findWords('4663', stubbed_dictionary));
```