

1. Write a function that returns True if a given name can generate an array of words.

Examples

```
anagram("Justin Bieber", ["injures", "ebb", "it"])    True
```

```
anagram("Natalie Portman", ["ornamental", "pita"])    True
```

```
anagram("Chris Pratt", ["chirps", "rat"])    False  
# Not all letters are used
```

```
anagram("Jeff Goldblum", ["jog", "meld", "bluffs"])    False  
# "s" does not exist in the original name
```

2. Given an array of users, each defined by an object with the following properties: name, score, reputation create a function that sorts the array to form the correct leaderboard.

The leaderboard takes into consideration the score of each user of course, but an emphasis is put on their reputation in the community, so to get the trueScore, you should add the reputation multiplied by 2 to the score.

Once you know the trueScore of each user, sort the array according to it in descending order.

Examples

```
leaderboards([  
  { "name": "a", "score": 100, "reputation": 20 },  
  { "name": "b", "score": 90, "reputation": 40 },  
  { "name": "c", "score": 115, "reputation": 30 },  
)  [  
  { "name": "c", "score": 115, "reputation": 30 }, # trueScore = 175  
  { "name": "b", "score": 90, "reputation": 40 }, # trueScore = 170  
  { "name": "a", "score": 100, "reputation": 20 } # trueScore = 140  
]
```

3. Create a function that, given a phrase and a number of letters guessed, returns a string with hyphens - for every letter of the phrase not guessed, and each letter guessed in place.

Examples

```
hangman("helicopter", ["o", "e", "s"])    "-e---o--e-"
```

```
hangman("tree", ["r", "t", "e"])    "tree"
```

```
hangman("Python rules", ["a", "n", "p", "r", "z"])    "P----n r----"
```

```
hangman("He's a very naughty boy!", ["e", "a", "y"])    "-e"- a -e-y -a----y -y!"
```

4. The Collatz sequence is as follows:

- Start with some given integer n.
- If it is even, the next number will be n divided by 2.
- If it is odd, multiply it by 3 and add 1 to make the next number.
- The sequence stops when it reaches 1.

According to the Collatz conjecture, it will always reach 1. If that's true, you can construct a finite sequence following the aforementioned method for any given integer.

Write a function that takes in an integer n and returns the highest integer in the corresponding Collatz sequence.

Examples

```
max_collatz(10)    16
# Collatz sequence: 10, 5, 16, 8, 4, 2, 1
```

```
max_collatz(32)    32
# Collatz sequence: 32, 16, 8, 4, 2, 1
```

```
max_collatz(85)    256
# Collatz sequence: 85, 256, 128, 64, 32, 16, 8, 4, 2, 1
```

5. Write a function that sorts a list of integers by their digit length in descending order, then settles ties by sorting numbers with the same digit length in ascending order.

Examples

```
digit_sort([77, 23, 5, 7, 101])
[101, 23, 77, 5, 7]
```

```
digit_sort([1, 5, 9, 2, 789, 563, 444])
[444, 563, 789, 1, 2, 5, 9]
```

```
digit_sort([53219, 3772, 564, 32, 1])
```

[53219, 3772, 564, 32, 1]