1 - add

Write a function that returns the sum of two numbers.

Example

For `param1 = 1` and `param2 = 2`, the output should be

`add(param1, param2) = 3`.

Input/Output

- **[execution time limit] 4 seconds (js)**
- **[input] integer param1**
  Guaranteed constraints:
  `-1000 ≤ param1 ≤ 1000`.
- **[input] integer param2**
  Guaranteed constraints:
  `-1000 ≤ param2 ≤ 1000`.
- **[output] integer**
  The sum of the two inputs.

```
function add(param1, param2) {

}
```

2 - centuryFromYear

Given a year, return the century it is in. The first century spans from the year 1 up to and including the year 100, the second - from the year 101 up to and including the year 200, etc.

Example

- For `year = 1905`, the output should be `centuryFromYear(year) = 20`;
- For `year = 1700`, the output should be `centuryFromYear(year) = 17`.

Input/Output

- **[execution time limit] 4 seconds (js)**
- **[input] integer year**
  A positive integer, designating the year.
  Guaranteed constraints:
  `1 ≤ year ≤ 2005`.
- **[output] integer**
  The number of the century the year is in.

```
function centuryFromYear(year) {

}
```

3 - checkPalindrome

Given the string, check if it is a palindrome. Palindrome is A string that doesn't change when reversed (it reads the same backward and forward).

Examples:

- `"eye"` is a palindrome
- `"noon"` is a palindrome
- `"decaf faced"` is a palindrome
- `"taco cat"` is **not** a palindrome (backwards it spells `"tac ocat"`)
- `"racecars"` is **not** a palindrome (backwards it spells `"sracecar"`)

Example

- For `inputString = "aabaa"`, the output should be `checkPalindrome(inputString) = true`;
- For `inputString = "abac"`, the output should be `checkPalindrome(inputString) = false`;
- For `inputString = "a"`, the output should be `checkPalindrome(inputString) = true`.

Input/Output

- **[execution time limit] 4 seconds (js)**
- **[input] string inputString**
  A non-empty string consisting of lowercase characters.
  Guaranteed constraints:
  $1 \leq$ `inputString.length` $\leq 10^5$.
- **[output] boolean**
  `true` if `inputString` is a palindrome, `false` otherwise.

```
function checkPalindrome(inputString) {

}
```

4 - isLucky

Ticket numbers usually consist of an even number of digits. A ticket number is considered *lucky* if the sum of the first half of the digits is equal to the sum of the second half.

Given a ticket number `n`, determine if it's *lucky* or not.

Example

- For `n = 1230`, the output should be `isLucky(n) = true`;
- For `n = 239017`, the output should be `isLucky(n) = false`.

Input/Output

- **[execution time limit] 4 seconds (rb)**
- **[input] integer n**
  A ticket number represented as a positive integer with an even number of digits.
  Guaranteed constraints:
  $10 \leq n < 10^6$.
- **[output] boolean**
  `true` if `n` is a lucky ticket number, `false` otherwise.

```
function isLucky(n) {

}
```

5 - isIPv4Address

An IP address is a numerical label assigned to each device (e.g., computer, printer) participating in a computer network that uses the Internet Protocol for communication. There are two versions of the Internet protocol, and thus two versions of addresses. One of them is the IPv4 address.

An identification number for devices connected to the internet. An IPv4 address written in dotted quad notation consists of four 8-bit integers separated by periods.

In other words, it's a string of four numbers each between `0` and `255` inclusive, with a `"."` character in between each number. All numbers should be present without leading zeros.

Examples:

- `192.168.0.1` is a valid IPv4 address
- `255.255.255.255` is a valid IPv4 address
- `280.100.92.101` is not a valid IPv4 address because `280` is too large to be an 8-bit integer (the largest 8-bit integer is `255`)
- `255.100.81.160.172` is not a valid IPv4 address because it contains 5 integers instead of 4
- `1..0.1` is not a valid IPv4 address because it's not properly formatted
- `17.233.00.131` and `17.233.01.131` are not valid IPv4 addresses because they contain leading zeros

Given a string, find out if it satisfies the IPv4 address naming rules.

Example

- For `inputString = "172.16.254.1"`, the output should be `isIPv4Address(inputString) = true`;
- For `inputString = "172.316.254.1"`, the output should be `isIPv4Address(inputString) = false`.
  `316` is not in range `[0, 255]`.
- For `inputString = ".254.255.0"`, the output should be `isIPv4Address(inputString) = false`.
  There is no first number.

Input/Output

- **[execution time limit] 4 seconds (js)**
- **[input] string inputString**
  A string consisting of digits, full stops and lowercase English letters.
  Guaranteed constraints:
  `1 ≤ inputString.length ≤ 30`.
- **[output] boolean**
  `true` if `inputString` satisfies the IPv4 address naming rules, `false` otherwise.

```
function isIPv4Address(inputString) {

}
```