Question 1:

Create a function that accepts a string, checks if it's a valid email address and returns either `true` or `false`, depending on the evaluation.

- The string must contain an `@` character.
- The string must contain a `.` character.
- The `@` must have at least one character in front of it.
    - e.g. `"e@edabit.com"` is valid while `"@edabit.com"` is invalid.
- The `.` and the `@` must be in the appropriate places.
    - e.g. `"hello.email@com"` is invalid while `"john.smith@email.com"` is valid.

If the string passes these tests, it's considered a valid email address.

Examples

```
ValidateEmail("@gmail.com") = false

ValidateEmail("hello.gmail@com") = false

ValidateEmail("gmail") = false

ValidateEmail("hello@gmail") = false

ValidateEmail("hello@edabit.com") = true
```

Notes

- Check the **Tests** tab to see exactly what's being evaluated.
- You can solve this challenge with RegEx, but it's intended to be solved with logic.
- Solutions using RegEx will be accepted but frowned upon :(

Question 2:

Joseph is in the middle of packing for a vacation. He's having a bit of trouble finding all of his socks, though.

Write a function that returns the number of sock pairs he has. A sock pair consists of two of the same letter, such as `"AA"`. The socks are represented as an unordered sequence.

Examples

```
SockPairs("AA") = 1

SockPairs("ABABC") = 2

SockPairs("CABBACCC") = 4
```

Notes

- If given an empty string (no socks in the drawer), return 0.
- There can be multiple pairs of the same type of sock, such as two pairs of cc for the last example.

Question 3:

Usually when you sign up for an account to buy something, your credit card number, phone number or answer to a secret question is partially obscured in some way. Since someone could look over your shoulder, you don't want that shown on your screen. Hence, the website masks these strings.

Your task is to create a function that takes a string, transforms all but the last four characters into "#" and returns the new masked string.

Examples

```
Maskify("4556364607935616") = "############5616"

Maskify("64607935616") = "#######5616"

Maskify("1") = "1"

Maskify("") = ""
```

Notes

- The maskify function must accept a string of any length.
- An empty string should return an empty string (fourth example above).

Question 4:

Write a function that accepts a string and an `n` and returns a cipher by rolling each character forward (`n > 0`) or backward (`n < 0`) **n** times.

Note: Think of the letters as a connected loop, so rolling `a` backwards once will yield `z`, and rolling `z` forward once will yield `a`. If you roll 26 times in either direction, you should end up back where you started.

Examples

```
RollingCipher("abcd", 1) = "bcde"

RollingCipher("abcd", -1) = "zabc"

RollingCipher("abcd", 3) = "defg"

RollingCipher("abcd", 26) = "abcd"
```

Notes

- All letters are lower cased.
- No spacing.
- Each character is rotated the same number of times.
- ASCII