字符反转

Q34 E01

reverse(text: String) -> String

要求：

将 text 反转

必须使用 递归

可以使用 substring

样例：

```
reverse("abc") -> "cba"
```

字符反转 高效版

Q34 E02

reverse(text: String) -> String

要求：

将 text 反转

必须使用 递归

不可以使用 substring

数列生成

Q33

介绍

我们可以根据以下规则生成一个很大的序列

1. 数列刚开始的时候 只有一个 0

2. 每回合 它都会将自己复制，并二进制取反叠加在现在的序列的后面

样例

```
回合1   0
回合2   01
回合3   0110
回合4   0110 1001
回合5   0110 1001 1001 0110
```

要求

number(k: Int) -> Int

要求：

返回 在上述生成规则中，能看到 第 k 个数的时候，的 第 k 个数是多少

k 从 1 开始

比如：：

```
number(1) -> 0
number(3) -> 1
```

表达式求值

Q32 E01

evaluate(expression: String) : String

要求：

对 expression 进行表达式求值

expression 内 由 数 和 运算符组成。

其中 数 一定是 个位数，且 没有 0

运算符 一定是 + - * / 之一

expression 中 可能存在空格

expression 案例

```
"1 + 2 / 3"
"3"
```

约束：

假设 expression 内 至少有一个数

表达式求值 多位数版

Q32 E02

evaluate(expression: String) : String

要求：

expression 中的数 可能是多位整数

expression 案例

```
12 + 2 * 3
```

表达式求值 优先级版

Q32 E03

evaluate(expression: String) : String

要求：

expression 中 可能存在 ()

expression 案例

```
(1 + 2) * 3
```

约束：

假设 () 如果存在，一定是合法的，不需要做合法性验证

提示：

recursion 递归调用（这里用不到递归思维）