Questions?
Task 1: Repeating Input

The following method repeats the input of the user (terminated by “end”):

```java
public static void repeatInput() {
    while(true) {
        String input = In.readLine();
        if (input.equals("end")) {
            return;
        }
        Out.println(input);
    }
}
```

> hello
> you
> end

- Rewrite the method in a recursive way
Task 2: Reversing the Input

Modify the recursive method such that it outputs all inputs after "end" was entered in a reverse fashion.
Iterative Power

The following method \texttt{poweri} returns the \textit{n}-th positive power \(x^n\) of a number \(x \in \mathbb{R}\) for \(n > 0\):

\[
\text{// pre: } n > 0 \\
\text{// post: return } x^n
\]

\texttt{public static double poweri(double x, int n)\{} \\
\hspace{1em} \texttt{double result = 1;} \\
\hspace{1em} \texttt{while (n > 0)\{} \\
\hspace{2em} \texttt{result *= x;} \\
\hspace{2em} \texttt{--n;} \\
\hspace{1em} \texttt{\}} \\
\hspace{1em} \texttt{return result;} \\
\texttt{\}}
Task 3: Recursive Power

- Implement a recursive method `powerr` to return the $n$-th positive power of a number $x \in \mathbb{R}$ for $n > 0$.

You may not use any loops (no `while`, no `for`, no `do-while`).
Task 4: Recursive Power Optimized

- Implement a recursive method `powers` to return the \( n \)-th positive power of a number \( x \in \mathbb{R} \) for \( n > 0 \) using a small number of multiplications.

Again, you may not use any loops (no `while`, no `for`, no `do-while`).

- Optional: improve the methods such that they can additionally deal with negative powers.