

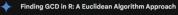


?

Please consider this programming problem:

Given two positive integers A and B, find GCD of A and B.

Provide an example solution for the problem in the R language.



- Understanding the Problem:
- GCD (Greatest Common Divisor): The largest positive integer that divides both A and B without leaving a remainder.
- Euclidean Algorithm: A classic algorithm for finding the GCD of two numbers. It's based on the observation that if GCD(A, B) = G, then GCD(B, A mod B) = G.

R Implementation:

```
Code-Snippet
gcd <- function(a, b) {
   if (b == 0) {
      return(a)
   } else {
      return(gcd(b, a %% b))
}
                                                                                                                                             0
Verwende den Code mit Vorsicht.
```

Explanation:

- 1. Base Case: If b is 0, then a is the GCD.
- 2. Recursive Call: Otherwise, recursively call $\ensuremath{\, \mbox{\sf gcd} \,}$ with $\ensuremath{\, \mbox{\sf b} \,}$ as the first argument and the remainder of a divided by b as the second argument.

Example Usage:

```
Code-Snippet
a <- 12
b <- 18
result <- gcd(a, b)
print(result) # Output: 6</pre>
Verwende den Code mit Vorsicht.
                                                                                                                0
```

How it works:

- For α = 12 and b = 18:
- The first recursive call is gcd(18, 12).
- The second recursive call is gcd(12, 6).
- The third recursive call is gcd(6, 0).
- Since b is 0, the function returns a, which is 6.

Advantages of this approach:

- Efficiency: The Euclidean algorithm is generally efficient, especially for large
- Clarity: The recursive implementation is often considered more readable and easier to understand than iterative versions.
- Versatility: The function can be easily adapted to other programming languages

This R implementation provides a clear and efficient solution for finding the GCD of two positive integers using the Euclidean algorithm.



