I'm trying to understand a given program for a R programming problem.
Can you help by explaining a specific part of this program? Below I first provide the problem description, then the program, and then a specific part of this program.

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

Program:

```
find_gcd <- function(A, B) {
 while (B != 0) {
  temp <- B
  B <- A %% B
  A <- temp
 }
 return(A)
}

A <- 56
B <- 98
gcd <- find_gcd(A, B)
cat("GCD of", A, "and", B, "is:", gcd, "\n")
```

Specific part:

```
while (B != 0) {
  temp <- B
  B <- A %% B
  A <- temp
 }
```

Can you provide a detailed explanation about the specific part above in the context of the whole program?

Show drafts

I'd be glad to provide a detailed explanation of the specific part of the R program in the context of the entire program:

Problem Statement:

> Given two positive integers A and B, find the GCD (Greatest Common Divisor) of A and B.

Program:

Code snippet

```
find_gcd <- function(A, B) {
  while (B != 0) {
    temp <- B
    B <- A %% B
    A <- temp
  }
  return(A)
}

A <- 56
B <- 98
gcd <- find_gcd(A, B)
cat("GCD of", A, "and", B, "is:", gcd, "\n")
```

Use code with caution.

Specific Part:

Code snippet

```
while (B != 0) {
  temp <- B
  B <- A %% B
  A <- temp
}
```

Use code with caution.

Explanation:

This loop is the core of the GCD algorithm, implementing the Euclidean algorithm. It iteratively calculates the GCD of A and B until B becomes 0. Here's a breakdown of what happens in each iteration:

1. `temp <- B` : This line temporarily stores the current value of B in the variable `temp` . This is crucial for the subsequent steps.
2. `B <- A %% B` : This line updates the value of B to the remainder of A divided by B. This is the key step in the Euclidean algorithm. The remainder represents the greatest common divisor of A and B at that point.
3. `A <- temp` : This line updates the value of A to the previously stored value of B (which is now in `temp`). This effectively swaps the values of A and B, preparing for the next iteration.

Overall Logic:

The loop continues to execute as long as B is not 0. In each iteration, it calculates the remainder of A divided by B and updates A and B accordingly. The process continues until B becomes 0, at which point the GCD is found in the value of A.

Example:

Let's consider A = 56 and B = 98:

1. **Iteration 1:**
   - `temp <- 98`
   - `B <- 56 %% 98 = 56`
   - `A <- 98`
2. **Iteration 2:**
   - `temp <- 56`
   - `B <- 98 %% 56 = 42`
   - `A <- 56`
3. **Iteration 3:**
   - `temp <- 42`
   - `B <- 56 %% 42 = 14`
   - `A <- 42`
4. **Iteration 4:**
   - `temp <- 14`
   - `B <- 42 %% 14 = 0`
   - `A <- 14`

Since B is now 0, the loop terminates, and the GCD is found in A, which is 14.

In essence, the specific part of the program effectively implements the Euclidean algorithm to find the GCD of two positive integers.