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

The specific part of the program you've highlighted, `while (B != 0) { ... }`, is the core of the Euclidean algorithm, which is used to find the greatest common divisor (GCD) of two numbers.

Here's a breakdown of what this loop does:

1. **Initialization:** The loop starts with the condition `B != 0`. This means that as long as B is not equal to zero, the loop will continue to execute.
2. **Temporary Storage:** Inside the loop, the value of B is temporarily stored in the variable `temp`. This is done because we'll be modifying the value of B in the next step.
3. **Remainder Calculation:** The value of B is updated to be the remainder of A divided by B. This is done using the modulo operator `%%`. In essence, we're finding out how many times B can fit into A without going over, and then storing the leftover part in B.
4. **Swap:** The value of A is updated to be the value of `temp`, which was the original value of B. This effectively swaps the values of A and B.

The overall effect of this loop is to repeatedly calculate the remainder of A divided by B and then swap the values of A and B until B becomes zero. When B becomes zero, it means that A has been reduced to the greatest common divisor of the original A and B.

To understand why this works, consider the following:

- If B is a divisor of A, then the remainder of A divided by B will be zero. In this case, the loop will terminate, and A will be the GCD.
- If B is not a divisor of A, then the remainder will be a smaller number than B. By swapping A and B, we essentially reduce the problem to finding the GCD of the smaller number and the remainder. This process continues until we reach a point where one of the numbers becomes zero.

In the context of the entire program, this loop is essential for finding the GCD of A and B. It efficiently calculates the GCD using the Euclidean algorithm, which is a well-known and efficient method for this task.