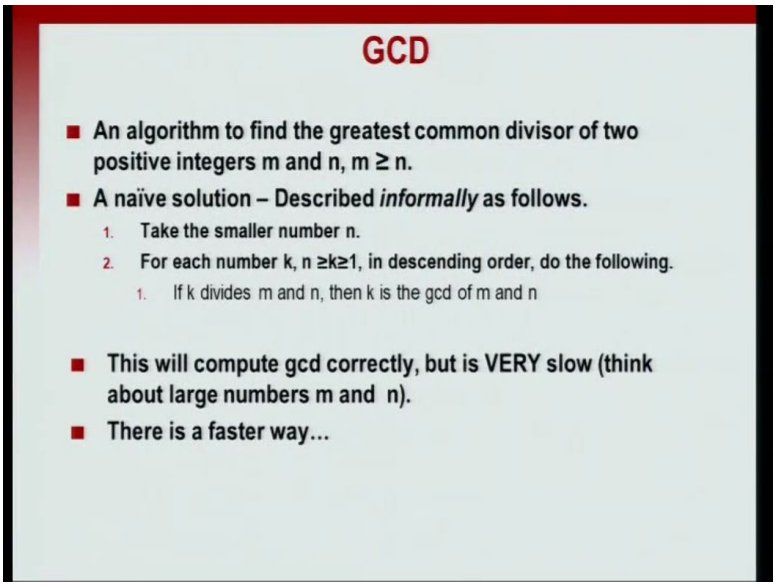
**Introduction to Programming in C**
**Department of Computer Science and Engineering**

In this session, we will write another algorithm to solve a mathematical problem. If you do not know this algorithm already, that is fine; it is more for the purpose of demonstrating, if you know a solution, how do you come up with the algorithm to tell a computer how to solve it.

(Refer Slide Time: 00:27)

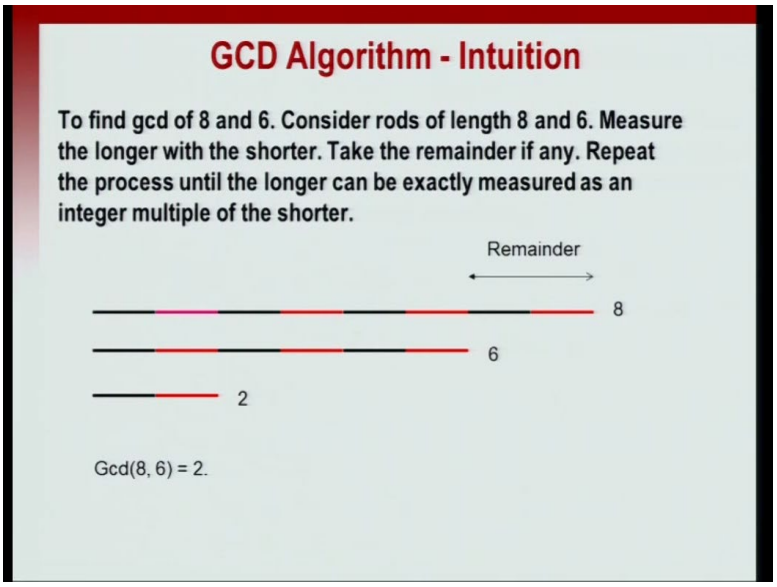



The algorithm is for finding the greatest common divisor or the highest common factor, this is known under two names of two positive integers: m and n. So, this is an algorithm you probably know. How do you solve this? Let us first try a naive solution. And before writing an algorithm, let us see what do I mean by the simple solution of GCD. So, you are asked to find the greatest common divisor of m and n; take the smaller number n; and now you start looking for each number k between 1 and n, remember that n is the smaller number; in descending order, do the following. What you do is if k divides m and n, then k is the greatest common divisor of m and n.

And this is obvious by the definition of greatest common divisor; if k divides m and n, then it is obviously a divisor of m and n. Also we are coming in descending order; we start from n and go down to n. So, the first divisor that you hit when you go down is going to be the greatest common divisor of m and n. So, this algorithm obviously works. It will compute the GCD correctly, but it is very slow. And think about a very large

numbers: m and n; and you will see that, it may go n steps before reaching the correct GCDs. So, compute the GCD of two very large numbers, which are relatively prime to each other; that means that the GCD of m and n are 1. Now, if you pick such a pair, this algorithm will compute the GCD correctly, but it will take n steps, because you have to go down all the way from n to 1 before you will hit the GCD. Can we do better? There is a faster way and it is a very old algorithm.

(Refer Slide Time: 02:40)

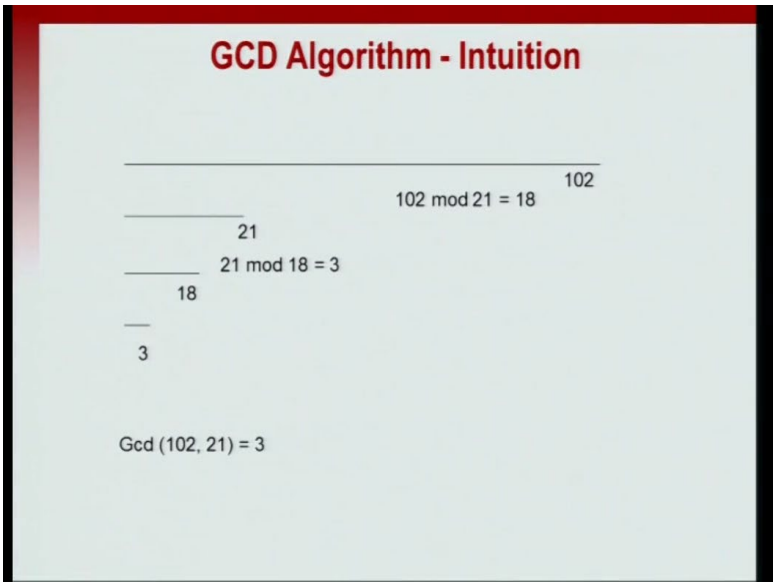



The algorithm is due to Euclid. We will see a slightly modified version of that algorithm. So, before we go into Euclid's algorithm for GCD, we will describe what it does and give you a slight intuition of why it works. So, consider the GCD of 8 and 6. Now, you can consider two rods: one of length 8, and another of length 6. Now, obviously, if a number divides 6 and 8, then I should be able to make a stick of that length, so that I can measure 6 exactly with that shorter rod; and I can measure 8 exactly with that shorter rod. This is the meaning of a common divisor, and we have to find the greatest common divisor.

So, first, what we will do is we will measure the longer rod using the shorter rod. Now, it may not measure the longer rod exactly. For example, in this case, 6 does not measure 8 exactly; there will be a small piece of length 2 left over. So, take that reminder. And now, repeat the process; now, 2 has become the shorter rod and 6 has become the longer rod. Now, see if 2 measures 6 exactly; it does. So, you are done. And then you can say that, 2

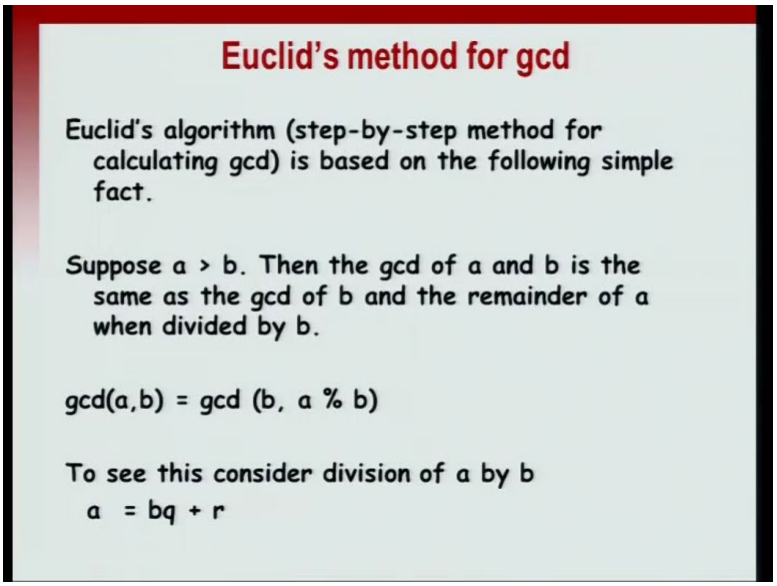
is the GCD of 8 and 6. The reason why this works is – by the nature of this algorithm, it is clear that 2 divides 6, because that is why we stop the algorithm. And also, we know that, 8 is basically 6 plus 2. So, it is obviously, a multiple of 2. So, it is a common divisor. And with a slightly more elaborate argument, we can argue that, it is the greatest common divisor. So, this is an algorithm, which is essentially due to Euclid. So, it was known for at least 2000 years.

(Refer Slide Time: 05:00)



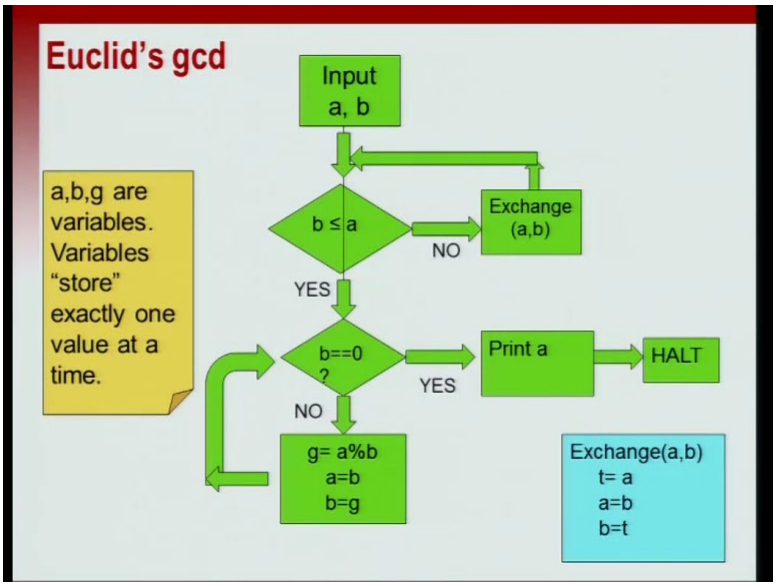

Let us pick a slightly more elaborate example. Let us say we want to find the GCD of 102 and 21. The process of taking remainder is what is known as the modulo operator in mathematics. So, 102 modulo 21 is the remainder of integer division of 102 by 21. So, the remainder of when you divide 102 by 21 is 18. So, that is the shorter rod for the next stage. Now, 21 mod 18 is 3. And that becomes the rod for the next stage; the shorter rod for the next stage. And 18 mod 3 is 0; that is when you stop the algorithm. So, when the modulo operator gives you 0 result; that means that, the shorter number exactly divides the larger number; that means that, the shorter number is a divisor of the larger number and you stop the algorithm. Now, you say that, GCD of 102 and 21 is 3.

(Refer Slide Time: 06:18)




So, this is a slight modification of the classical Euclid's method for GCD. And so, it is based on the following simple fact, which we have described. And you can prove this mathematically as well. So, suppose you take two positive numbers: a and b; where, a is the larger number; then GCD of a and b is the same as GCD of b and the remainder when you divide a by b. So, it is written by the equation GCD(a, b) is GCD(b, a % b). The modulo operator is represented as the percentage sign, because this is the convention that we will use in C. And this equation can be seen by our previous slide; a was the bigger rod; b was the shorter rod. This was the first stage. The second stage was when b is the shorter rod. And the shorter rod for the next stage is modulo – is given by the modulo operator. To prove this, you can start by considering the division of a by b and writing a as bq + r. But, we will not go into the proof. From elementary properties of natural numbers, it is possible to prove that, Euclid's method correctly computes the GCD.

(Refer Slide Time: 07:48)

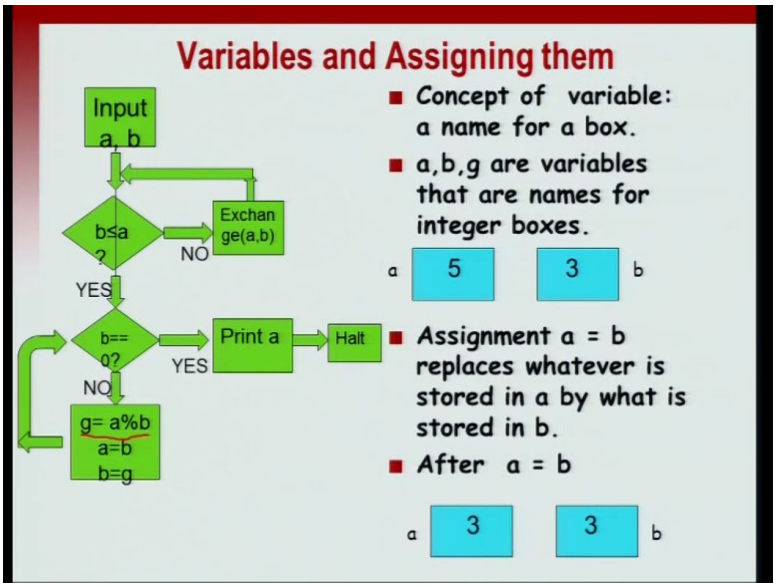



Right now, we will move into how do we write the GCD algorithm in the form of an input. So, here is a slightly abbreviated picture. I have skipped the start state; but the start state is there. Let us focus on what happens during the algorithm. You have two numbers: a and b. The first thing to ensure is that, a is the larger number. The reason we do that is that, if a is the larger number, then the modulo operator is properly defined. So, if a is the larger number, then we are fine; we can go into the GCD algorithm. If a is not the larger number, you merely swap a and b, so that whatever is the larger number, you called it a. So, exchange a and b; means that you say that, the value of a is stored in a temporary variable; then the value of b is stored in a; and then the value of b is stored in t with the value of t stored in b. So, here is a way to exchange the values of a and b. So, ensure at first that, a is the larger number. Once you do that, you get into the code for the proper utility in GCD.

First you test whether b is 0. If b is 0, then there is nothing to do; a is the GCD of a and b; GCD(4, 0) is 4; GCD(4, 0) is 8, and so on. So, if the smaller number is 0, then there is nothing do in the algorithm; the algorithm is over; and you say that, print a. If b is not 0, then we do the Euclidean equation. You take a % b; store it in a variable g; then assign the value of b to a and assign the value of g to b. So, this corresponds to the operation of taking b and a % b as the next step. After you do that, you again test the condition whether b has now become 0. If it is 0, then we are done and a is the GCD; otherwise, we do another round of taking a % b and setting a = b and b = g. So, a, b and g are what are

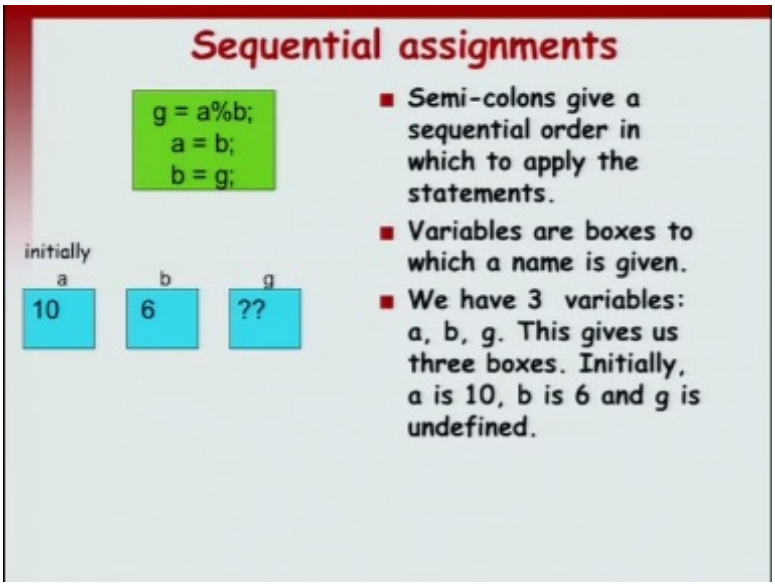
known as variables. And variables are used in programming to store exactly one value at a time. So, at any particular time, it will have one value; then after the execution of another instruction, it will have a new value and so on.

(Refer Slide Time: 10:43)



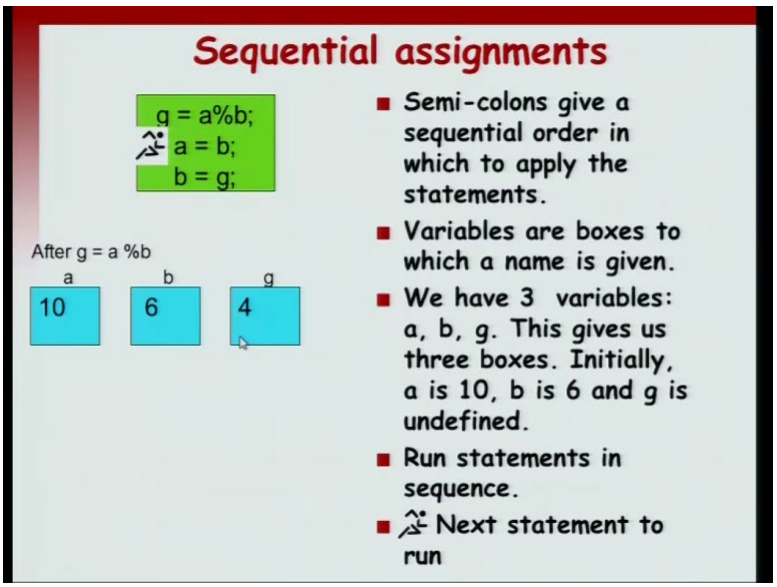

Now, for the purposes of describing an algorithm, imagine that, the variable is a box; and it is a name of a box; and the value is stored inside the box. For example, a, b and g are the variables that we have used in the program. And they are the names for these integer boxes. So, if we are computing, let us say the GCD of 5 and 3, then you might start with a equal to 5 and b equal to 3. The second operation that we have used in the code is the assignment operation. So, this is what an example of the assignment operator. And when we do an assignment, what we mean is that, you take the left variable, which is g in this case and assign it the value of what is the expression on the right-hand side, which is a % b . So, assignment a = b replaces whatever is stored in a by what is stored in b. So, take the right-hand side; take the value of that; and put it into the variable that the left-hand side represents. For example, if a was 5 and b is 3; after a = b, you would take the value of b and put it in a. So, a will now become 3 and b will remain 3.

(Refer Slide Time: 12:17)



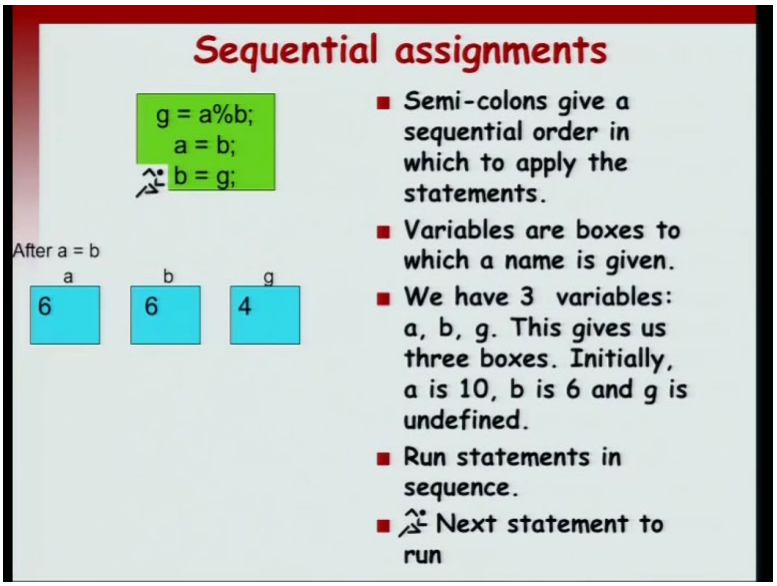

Another small thing that we have used in the code is sequential assignment. So, if you write a bunch of statements one after the other, let us say separated by semicolons; then this means that, the instructions are to be executed one after the other in sequence. So, first, you do g equal to a % b; then you do a = b; and after that you do b = g. So, initially, let us say that a is 10 and b is 6; g is undefined.

(Refer Slide Time: 12:50)



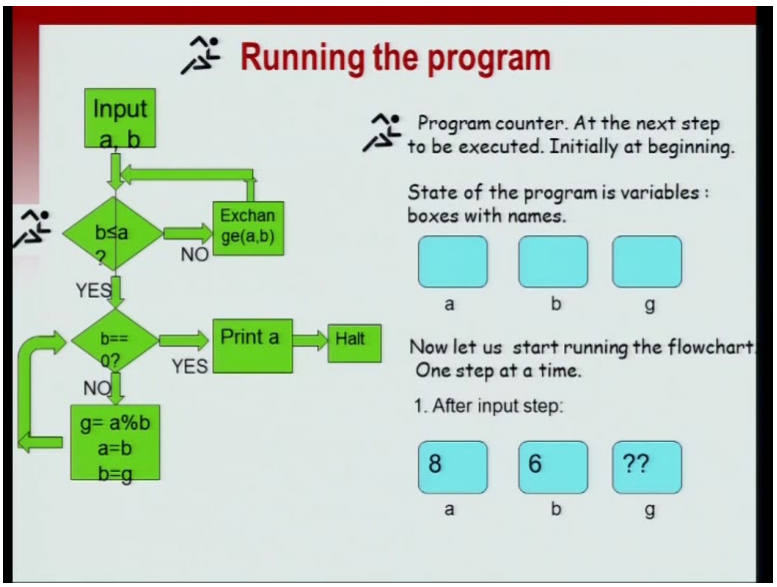

After you run the statement g equal to a % b, you take 10 modulo 6; you will have 4.

(Refer Slide Time: 12:59)




And then a = b; the value of b will be stored in a. So, a become 6. And then b = g; the value of g will be stored in b. So, b will become 4.

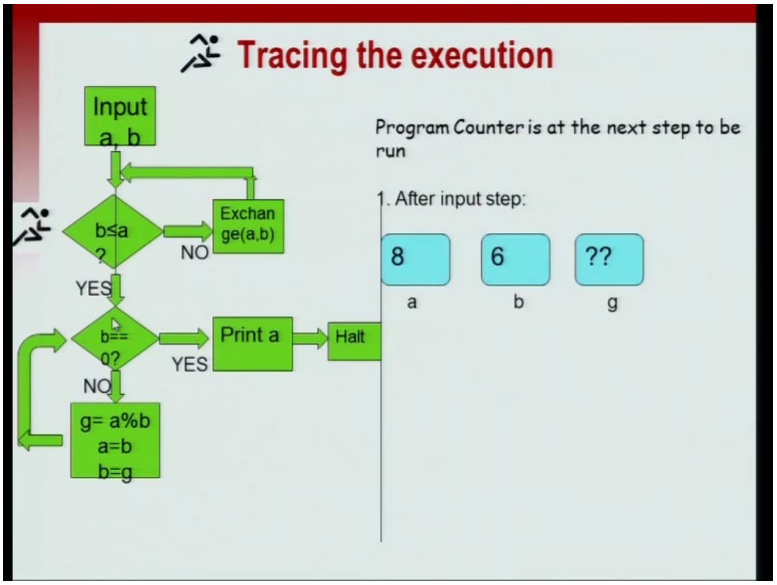
(Refer Slide Time: 13:12)




Now, let us just dry run the program or the algorithm and see how it computes the GCD of two numbers. So, I will denote the currently executing statement with an icon and I will call this the program counter. So, this is at any point, it is the next step to be executed. Initially, it is at the beginning of the code; where, you take the input. And we will have three variables, which will represent the current state of a program. So, suppose

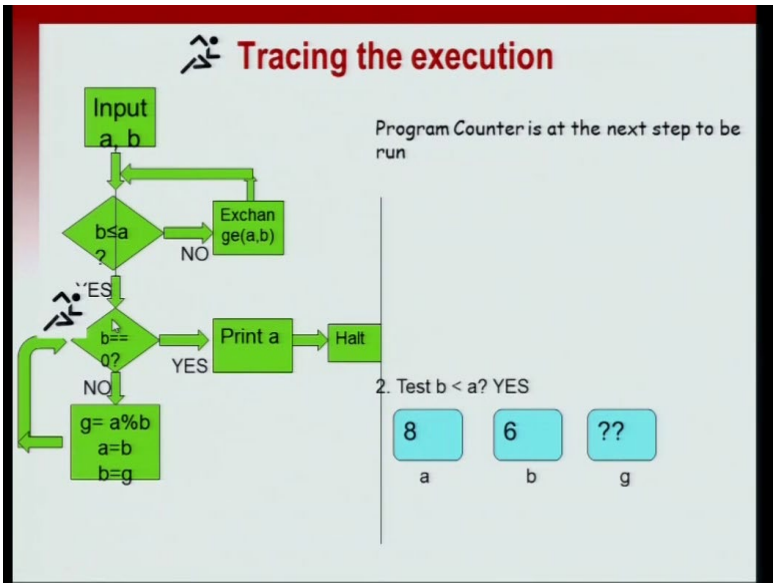
you want to compute the GCD of 8 and 6. So, you have a equal to 8; b equal to 6. You know that a is greater than b.

(Refer Slide Time: 14:02)



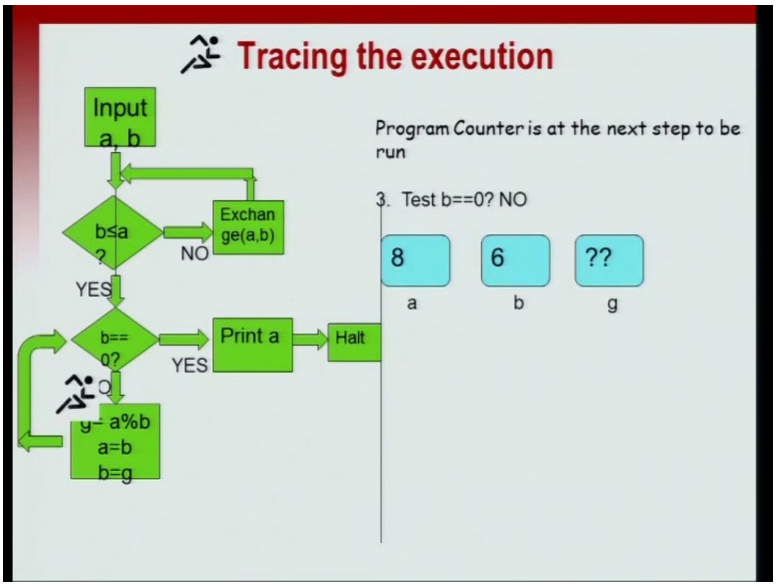

So, you proceed.

(Refer Slide Time: 14:09)



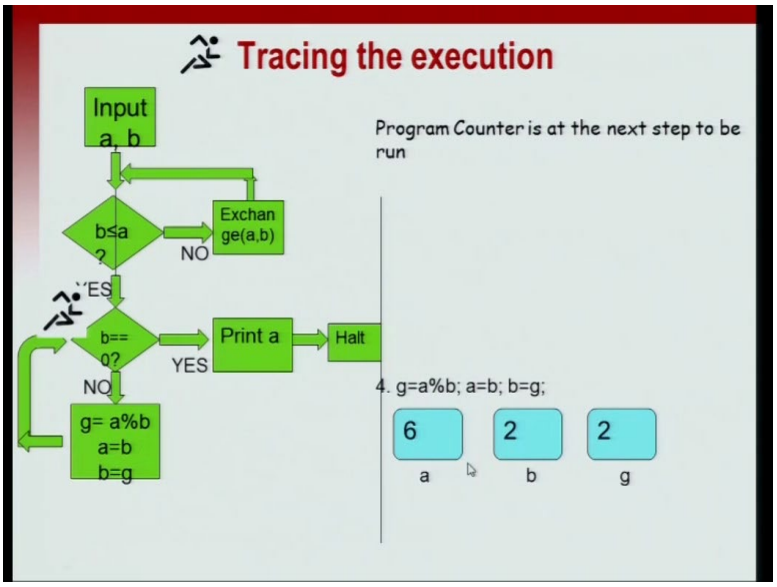

Now, you test whether b is 0.

(Refer Slide Time: 14:17)



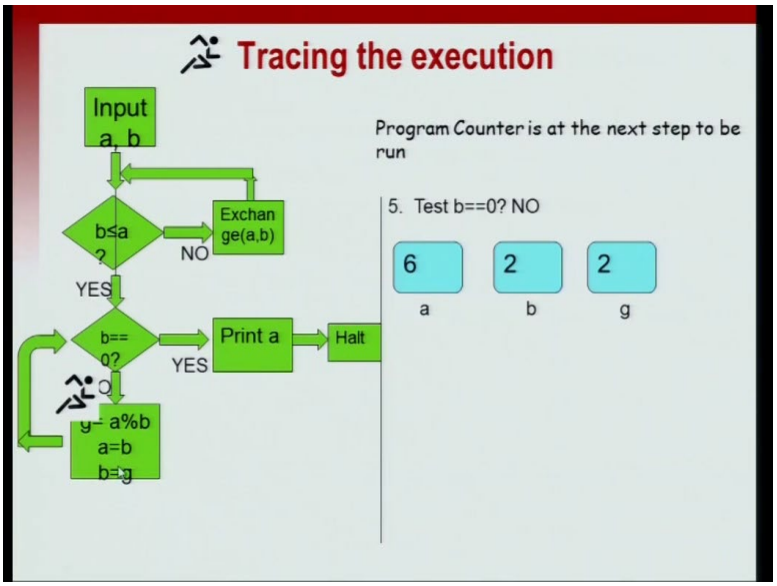

So, since b is non zero, you go into the main body of the loop.

(Refer Slide Time: 14:22)



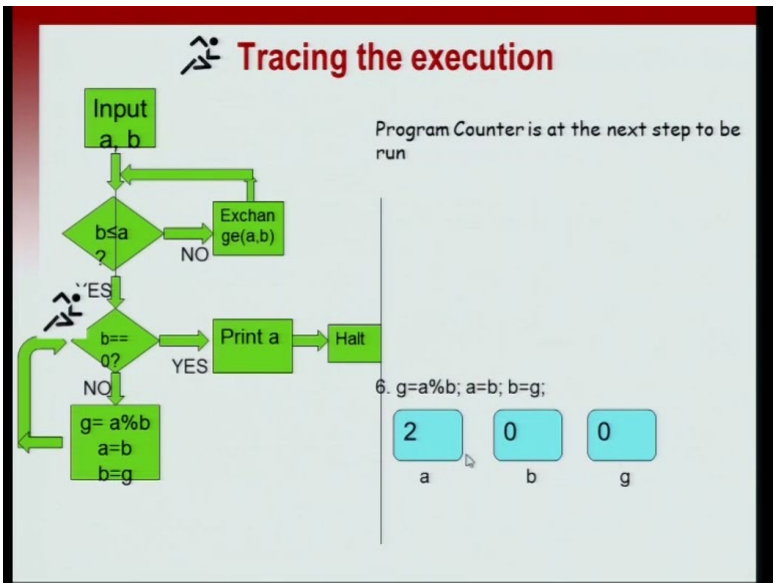

So, you do g equal to a % b; a = b; b = g, this step once. So, you will end up with a is now 6; b is 2; and g is 2. You again comeback to the discussion and test whether b is 0 or not; b is not 0.

(Refer Slide Time: 14:44)



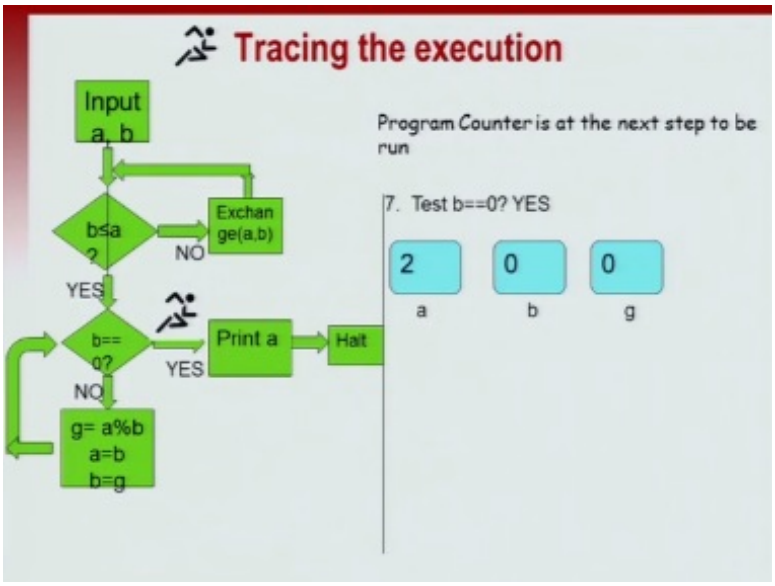

So, you go back into the body of the loop again. So, you have g to be a % b. So, 6 modulo 2 should be 0.

(Refer Slide Time: 14:58)



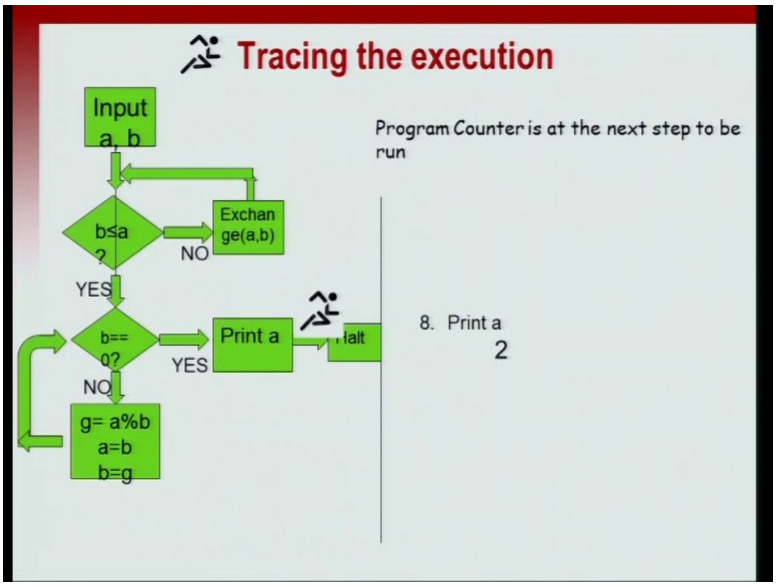

And then you do a = b and b = g. You will have a equal to 2; b equal to 0; and g equal to 0. At this point, b is now 0.

(Refer Slide Time: 15:20)




So, you say that a is actually the GCD of three numbers – of the numbers 8 and 6.

(Refer Slide Time: 15:22)




So, you can ensure that, it computes the GCD correctly.