Intro - GCD

Week1

GCD

- An algorithm to find the greatest common divisor of two positive integers m and n, $m \geq n$.

GCD

- An algorithm to find the greatest common divisor of two positive integers m and n, $m \geq n$.
- A naïve solution – Described *informally* as follows.

GCD

- An algorithm to find the greatest common divisor of two positive integers m and n, m ≥ n.
- A naïve solution – Described *informally* as follows.
  1. Take the smaller number n.

GCD

- **An algorithm to find the greatest common divisor of two positive integers m and n, m ≥ n.**
- **A naïve solution – Described *informally* as follows.**
  1. **Take the smaller number n.**
  2. **For each number k, n ≥k≥1, in descending order, do the following.**
     1. If k divides m and n, then k is the gcd of m and n

GCD

- **An algorithm to find the greatest common divisor of two positive integers m and n, m ≥ n.**
- **A naïve solution – Described *informally* as follows.**
  1. **Take the smaller number n.**
  2. **For each number k, n ≥k≥1, in descending order, do the following.**
     1. If k divides m and n, then k is the gcd of m and n

- **This will compute gcd correctly, but is VERY slow (think about large numbers m and n).**

GCD

- **An algorithm to find the greatest common divisor of two positive integers m and n, m ≥ n.**
- **A naïve solution – Described *informally* as follows.**
  1. **Take the smaller number n.**
  2. **For each number k, n ≥k≥1, in descending order, do the following.**
     1. If k divides m and n, then k is the gcd of m and n

- **This will compute gcd correctly, but is VERY slow (think about large numbers m and n).**
- **There is a faster way…**

GCD Algorithm - Intuition

GCD Algorithm - Intuition

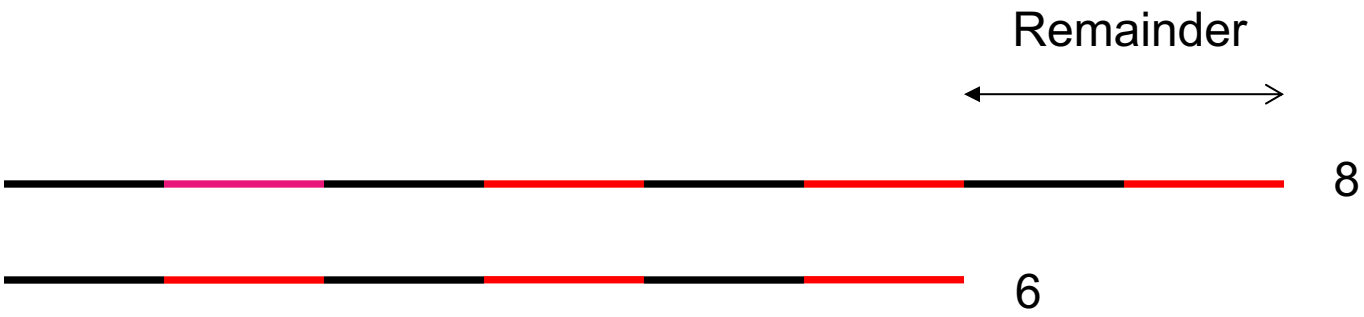
To find gcd of 8 and 6. Consider rods of length 8 and 6. Measure the longer with the shorter. Take the remainder if any. Repeat the process until the longer can be exactly measured as an integer multiple of the shorter.

GCD Algorithm - Intuition

To find gcd of 8 and 6. Consider rods of length 8 and 6. Measure the longer with the shorter. Take the remainder if any. Repeat the process until the longer can be exactly measured as an integer multiple of the shorter.

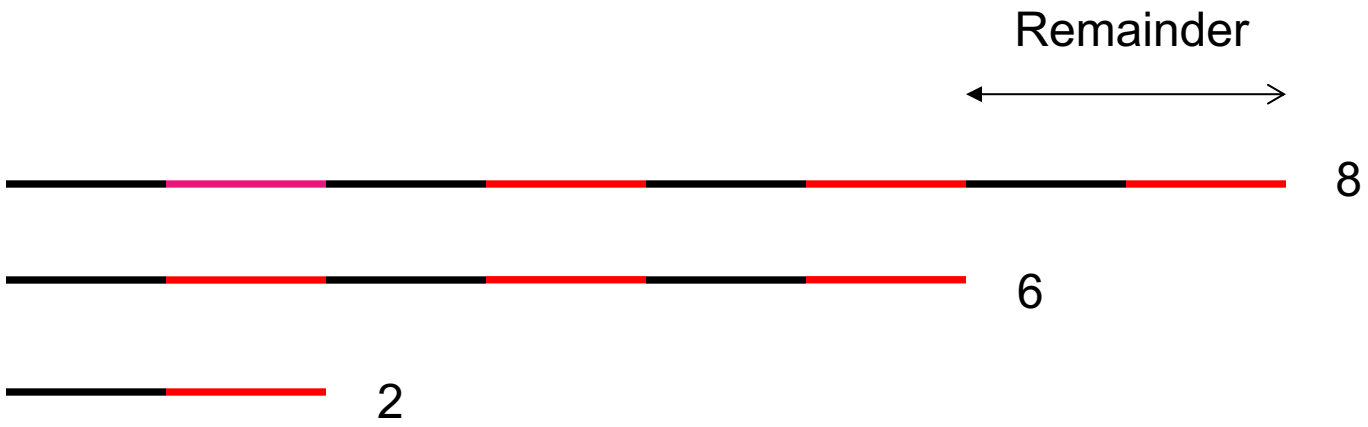
GCD Algorithm - Intuition

To find gcd of 8 and 6. Consider rods of length 8 and 6. Measure the longer with the shorter. Take the remainder if any. Repeat the process until the longer can be exactly measured as an integer multiple of the shorter.



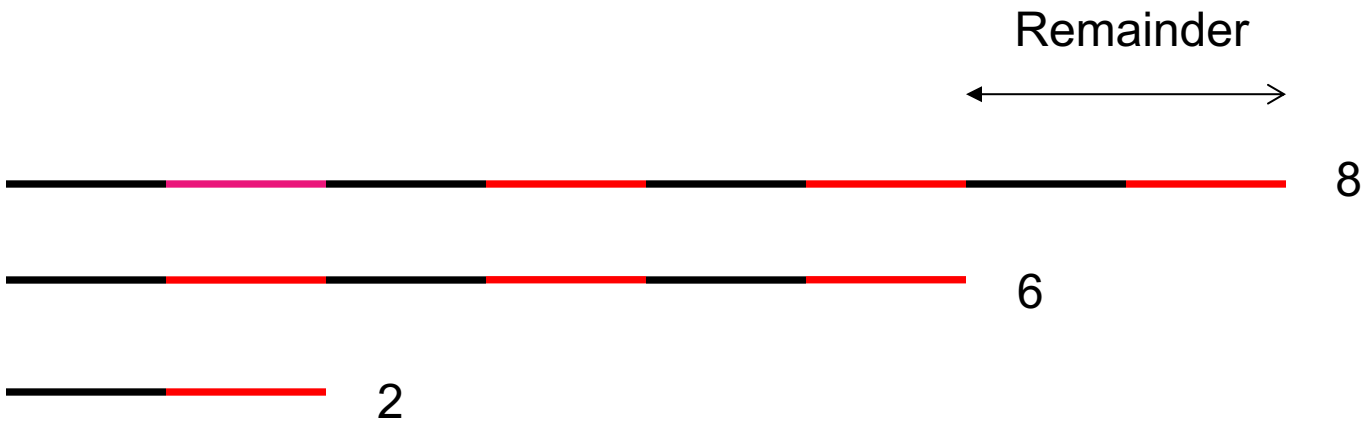
GCD Algorithm - Intuition

To find gcd of 8 and 6. Consider rods of length 8 and 6. Measure the longer with the shorter. Take the remainder if any. Repeat the process until the longer can be exactly measured as an integer multiple of the shorter.



GCD Algorithm - Intuition

To find gcd of 8 and 6. Consider rods of length 8 and 6. Measure the longer with the shorter. Take the remainder if any. Repeat the process until the longer can be exactly measured as an integer multiple of the shorter.



Gcd(8, 6) = 2.

GCD Algorithm - Intuition

GCD Algorithm - Intuition

102

21

GCD Algorithm - Intuition

102

102 mod 21 = 18

21

18

GCD Algorithm - Intuition

102

102 mod 21 = 18

21

21 mod 18 = 3

18

3

Gcd (102, 21) = 3

Euclid's method for gcd

Euclid's algorithm (step-by-step method for calculating gcd) is based on the following simple fact.

Suppose a > b. Then the gcd of a and b is the same as the gcd of b and the remainder of a when divided by b.

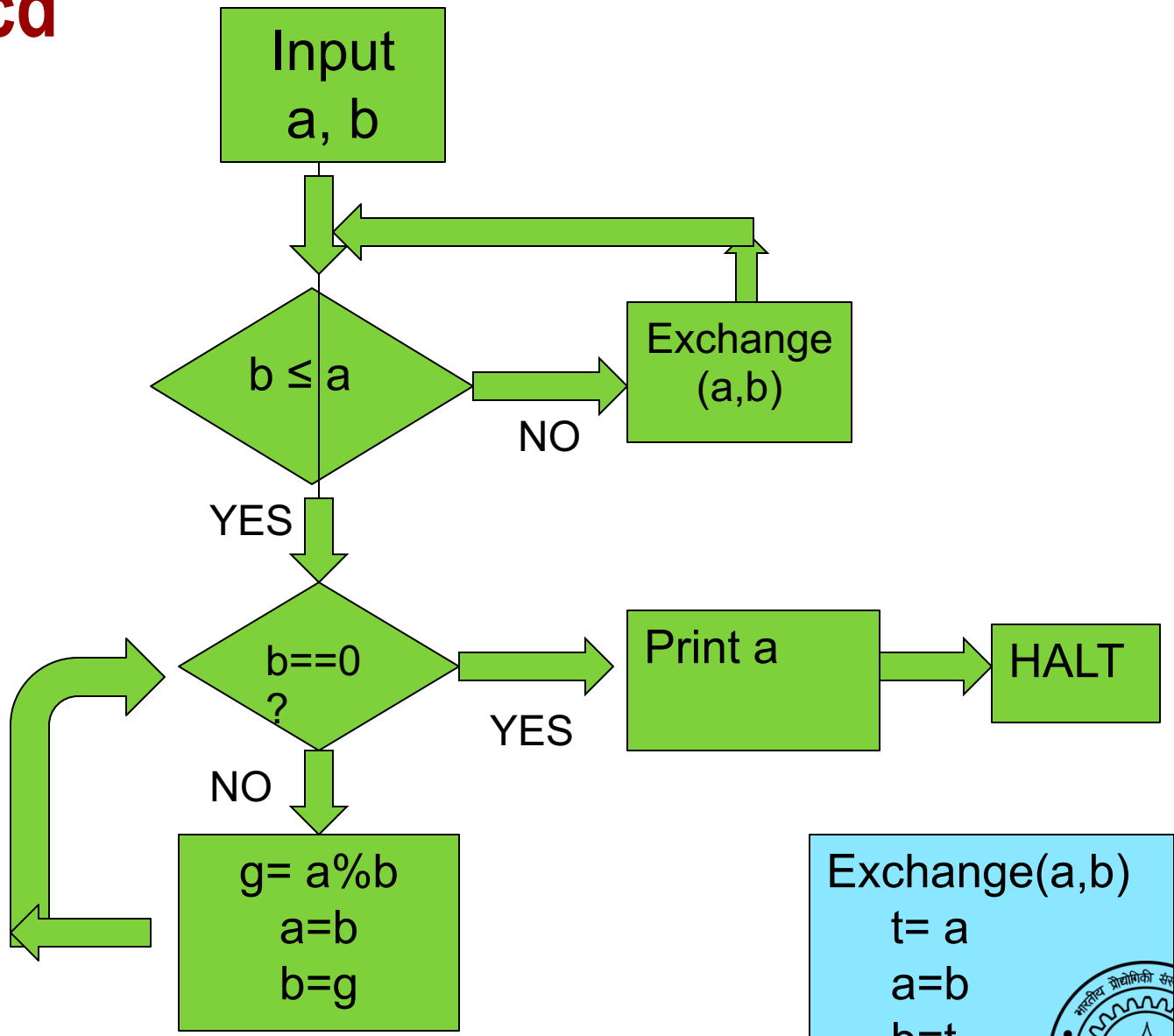
gcd(a,b) = gcd (b, a % b)

To see this consider division of a by b

a = bq + r

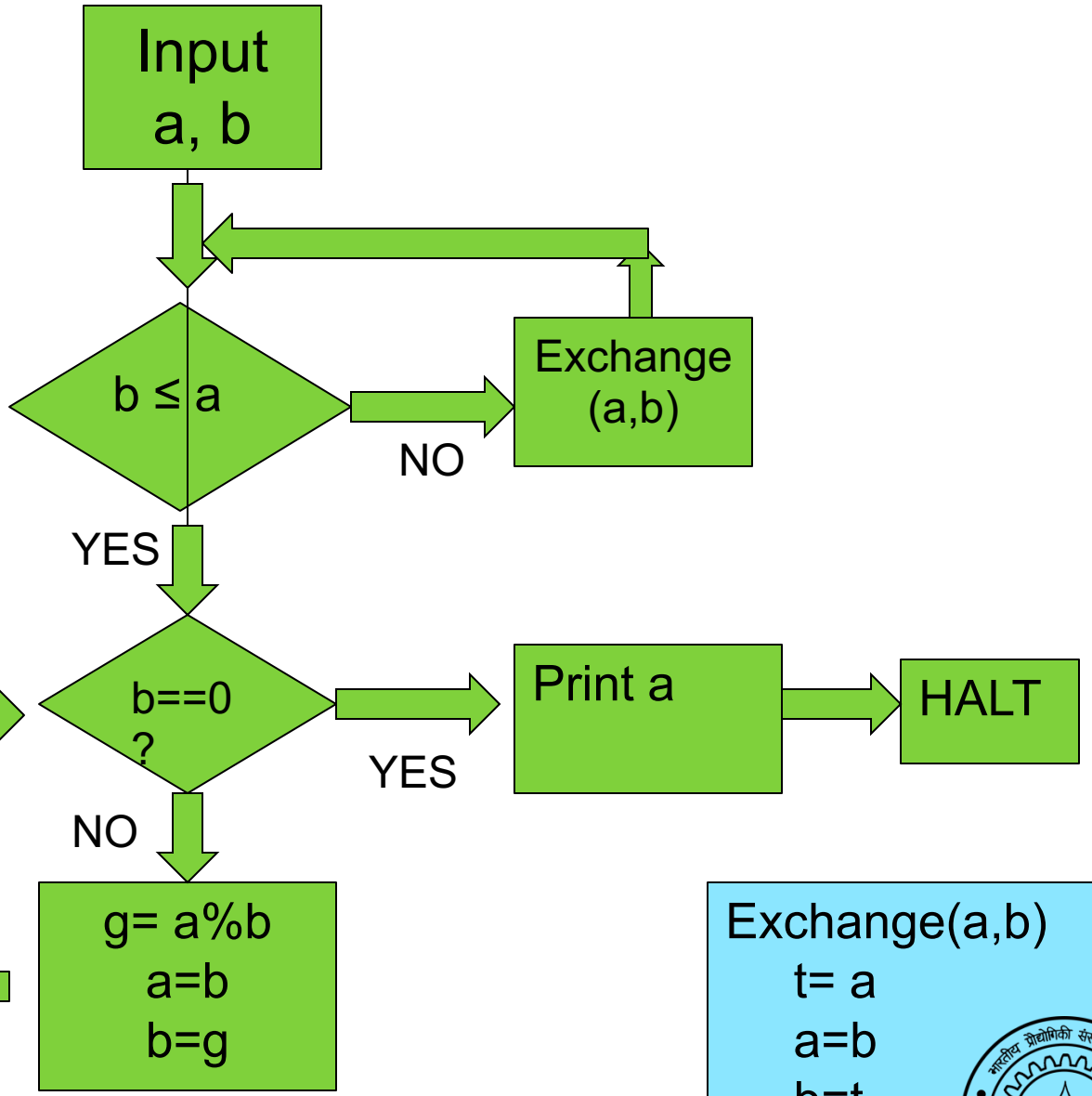
Euclid's gcd

a,b,g are variables. Variables "store" exactly one value at a time.



Exchange(a,b)
t= a
a=b
b=t

Euclid's gcd

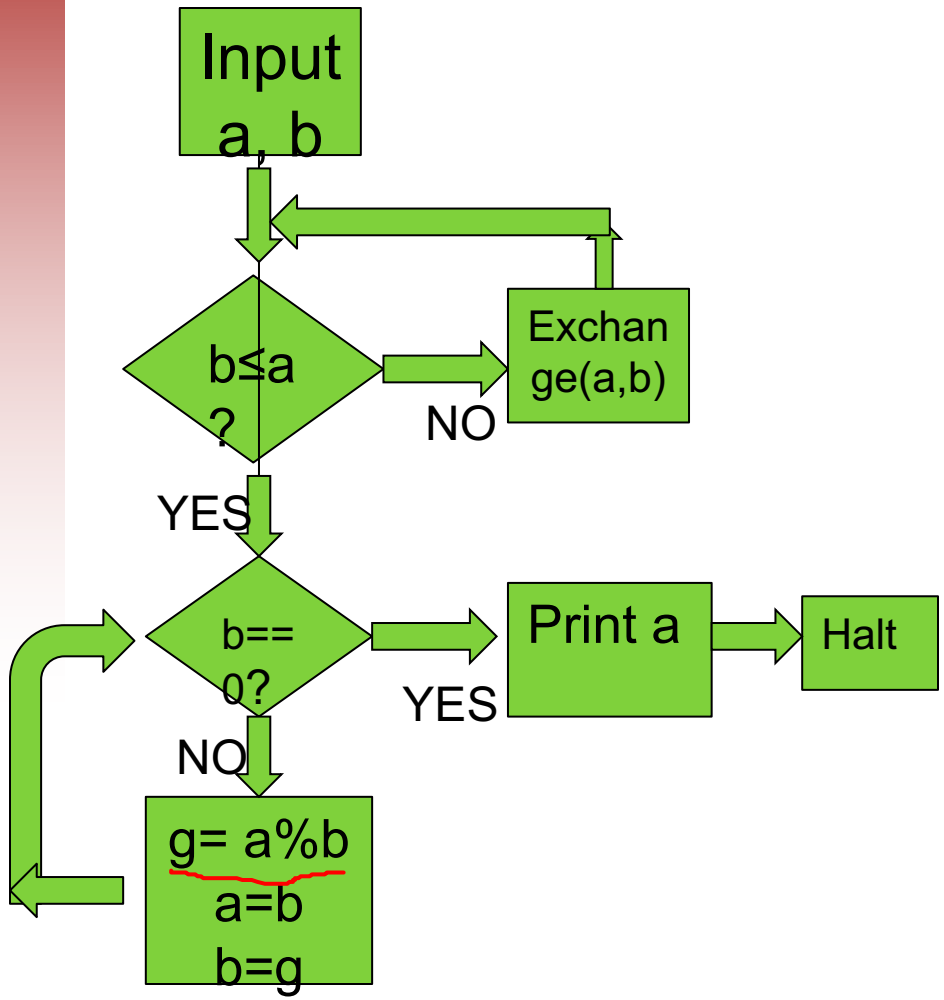


a,b,g are variables. Variables "store" exactly one value at a time.

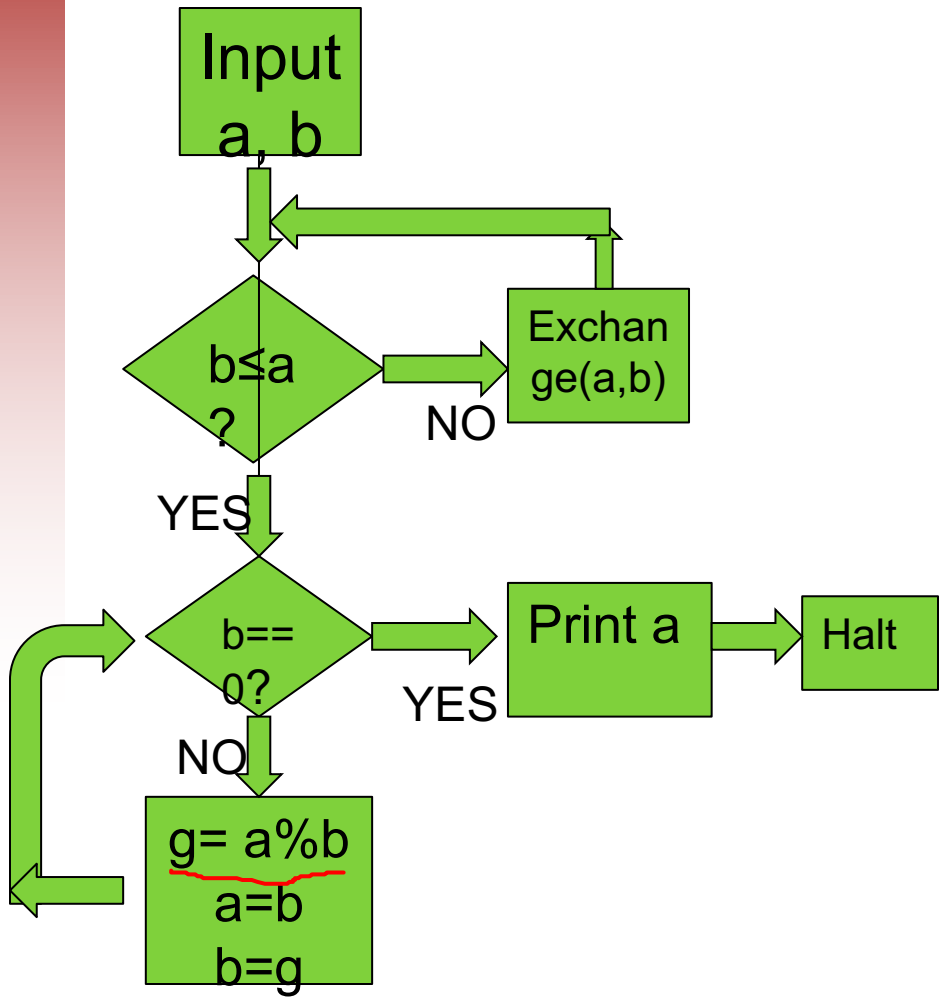
a%b is the remainder when a is divided by b.
Eg. 8%3 is 2

Exchange(a,b)
t= a
a=b
b=t

Variables and Assigning them

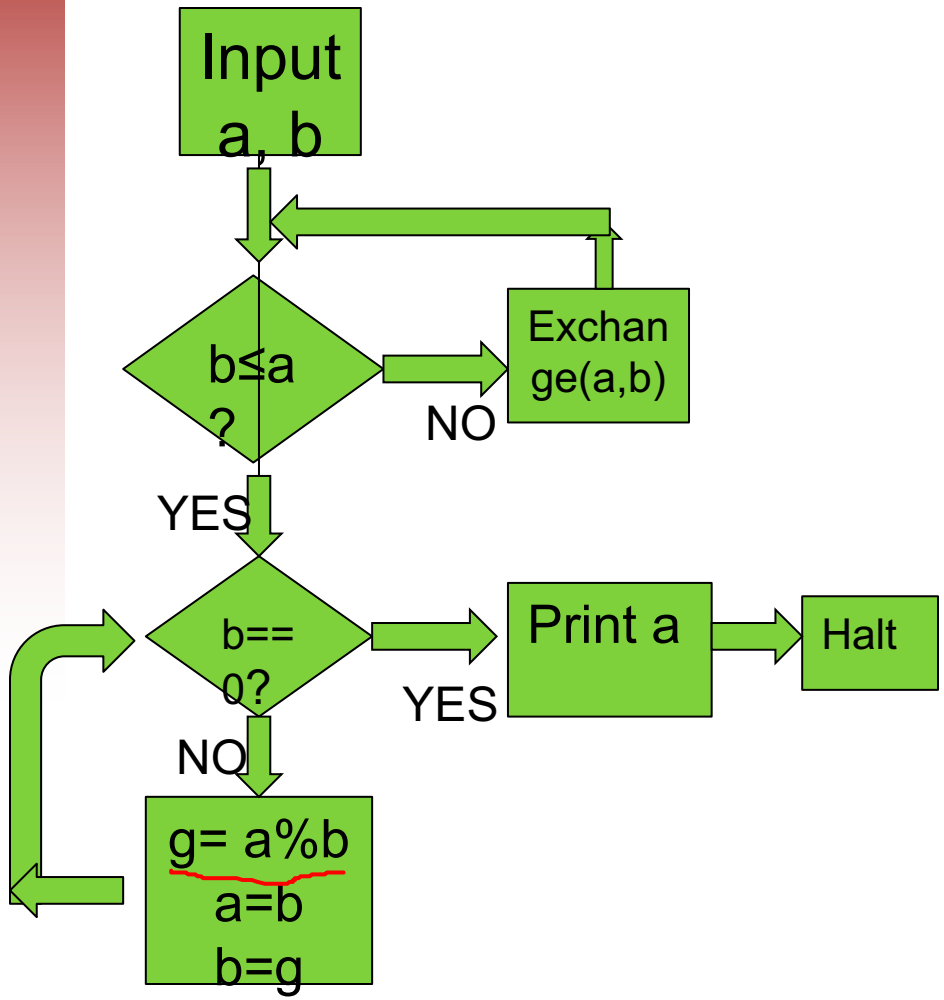


Variables and Assigning them



- Concept of variable: a name for a box.

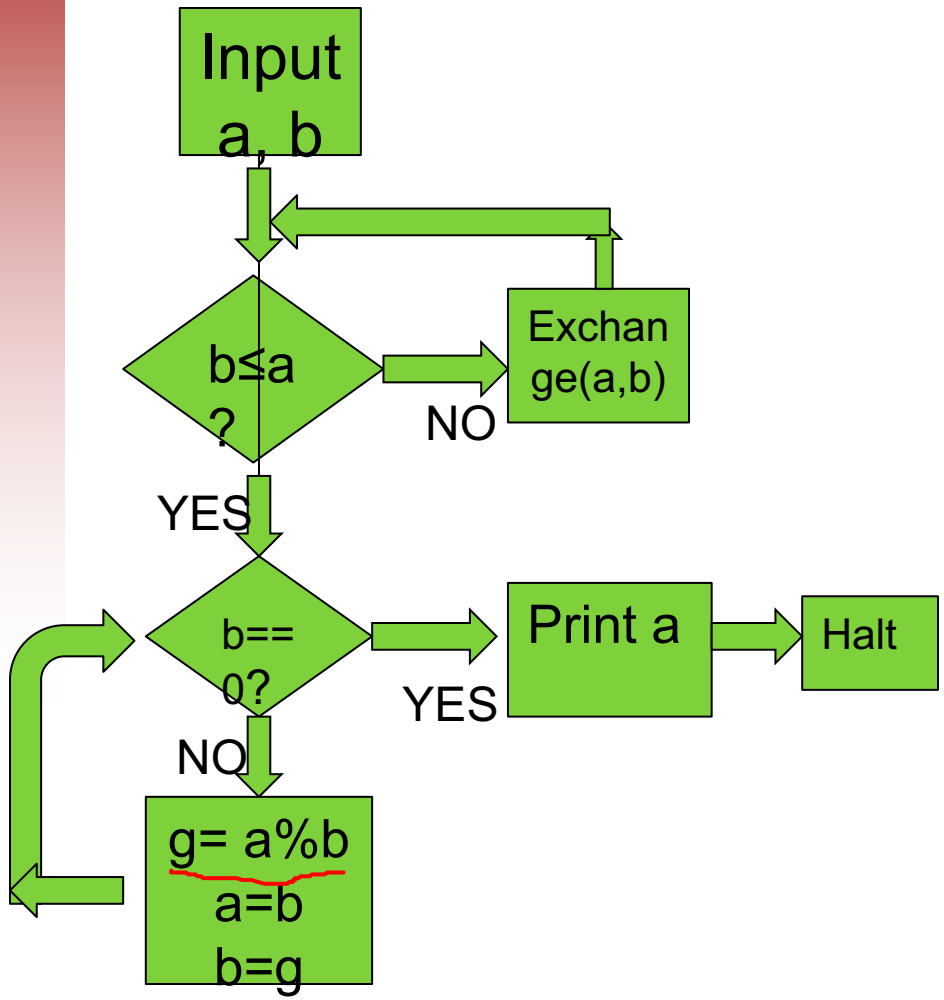
Variables and Assigning them



- Concept of variable: a name for a box.
- a,b,g are variables that are names for integer boxes.

a 5 3 b

Variables and Assigning them

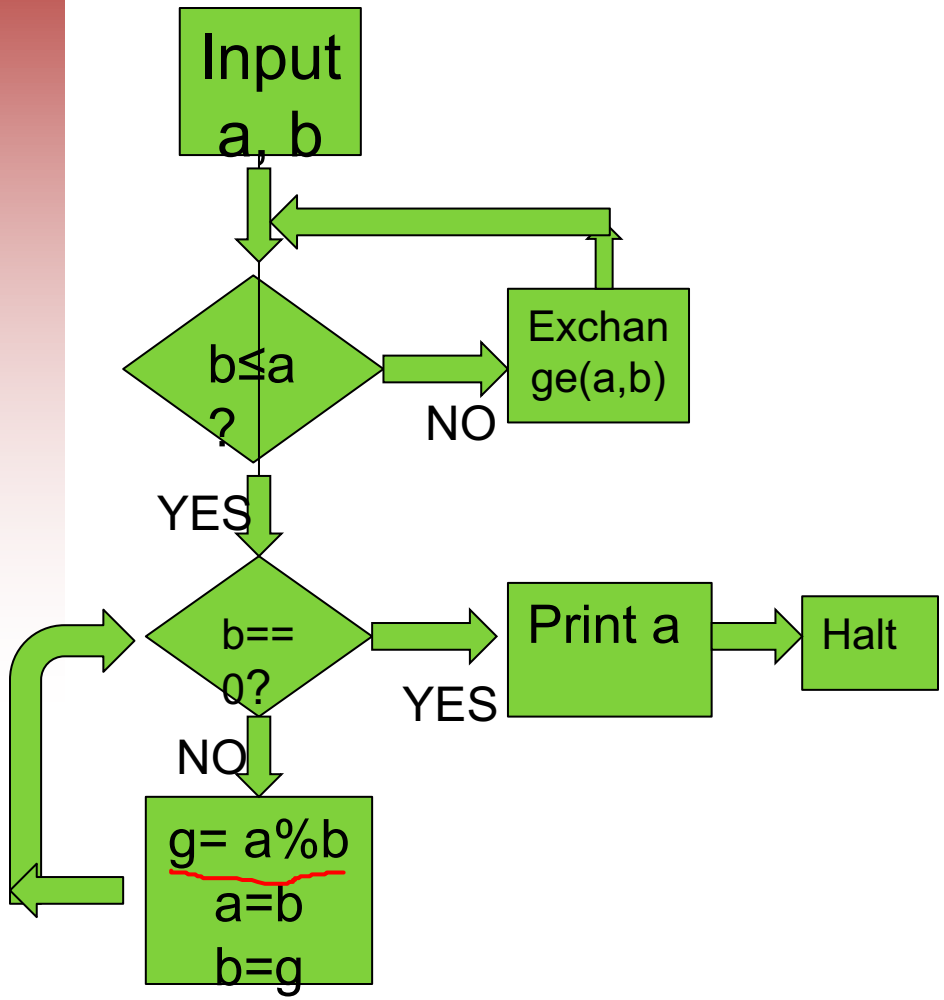


- Concept of variable: a name for a box.
- a,b,g are variables that are names for integer boxes.

a 5 3 b

- Assignment a = b replaces whatever is stored in a by what is stored in b.

Variables and Assigning them



- Concept of variable: a name for a box.
- a,b,g are variables that are names for integer boxes.

a | 5 | 3 | b

- Assignment a = b replaces whatever is stored in a by what is stored in b.
- After a = b

a | 3 | 3 | b

Sequential assignments

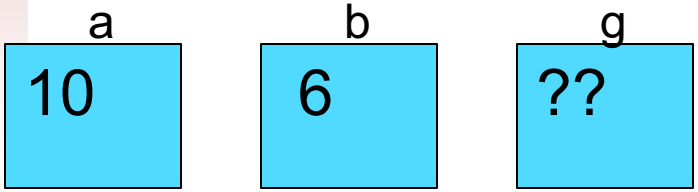
```
g = a%b;
a = b;
b = g;
```

- Semi-colons give a sequential order in which to apply the statements.

Sequential assignments

```
g = a%b;
a = b;
b = g;
```

initially

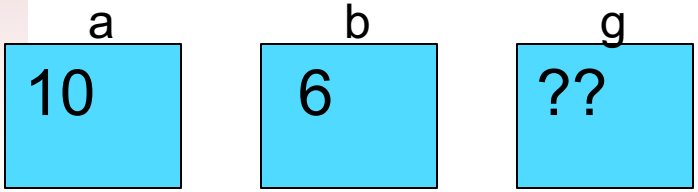


- Semi-colons give a sequential order in which to apply the statements.
- Variables are boxes to which a name is given.
- We have 3 variables: a, b, g. This gives us three boxes. Initially, a is 10, b is 6 and g is undefined.

Sequential assignments

```
g = a%b;
a = b;
b = g;
```

initially

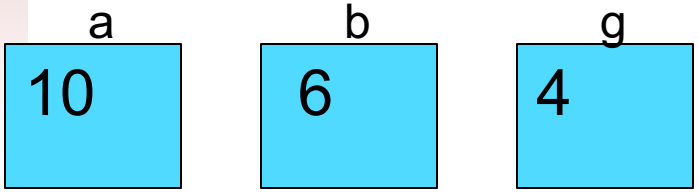


- Semi-colons give a sequential order in which to apply the statements.
- Variables are boxes to which a name is given.
- We have 3 variables: a, b, g. This gives us three boxes. Initially, a is 10, b is 6 and g is undefined.
- Run statements in sequence.
- Next statement to run

Sequential assignments

```
g = a%b;
a = b;
b = g;
```

After g = a %b

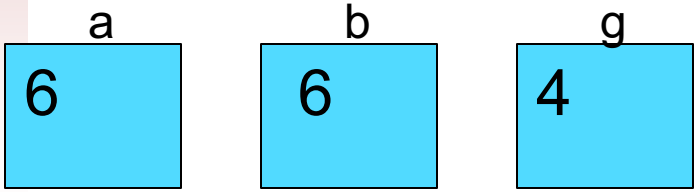


- Semi-colons give a sequential order in which to apply the statements.
- Variables are boxes to which a name is given.
- We have 3 variables: a, b, g. This gives us three boxes. Initially, a is 10, b is 6 and g is undefined.
- Run statements in sequence.
- Next statement to run

Sequential assignments

```
g = a%b;
a = b;
b = g;
```

After a = b

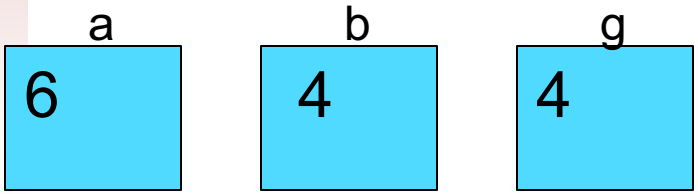


- Semi-colons give a sequential order in which to apply the statements.
- Variables are boxes to which a name is given.
- We have 3 variables: a, b, g. This gives us three boxes. Initially, a is 10, b is 6 and g is undefined.
- Run statements in sequence.
- Next statement to run

Sequential assignments

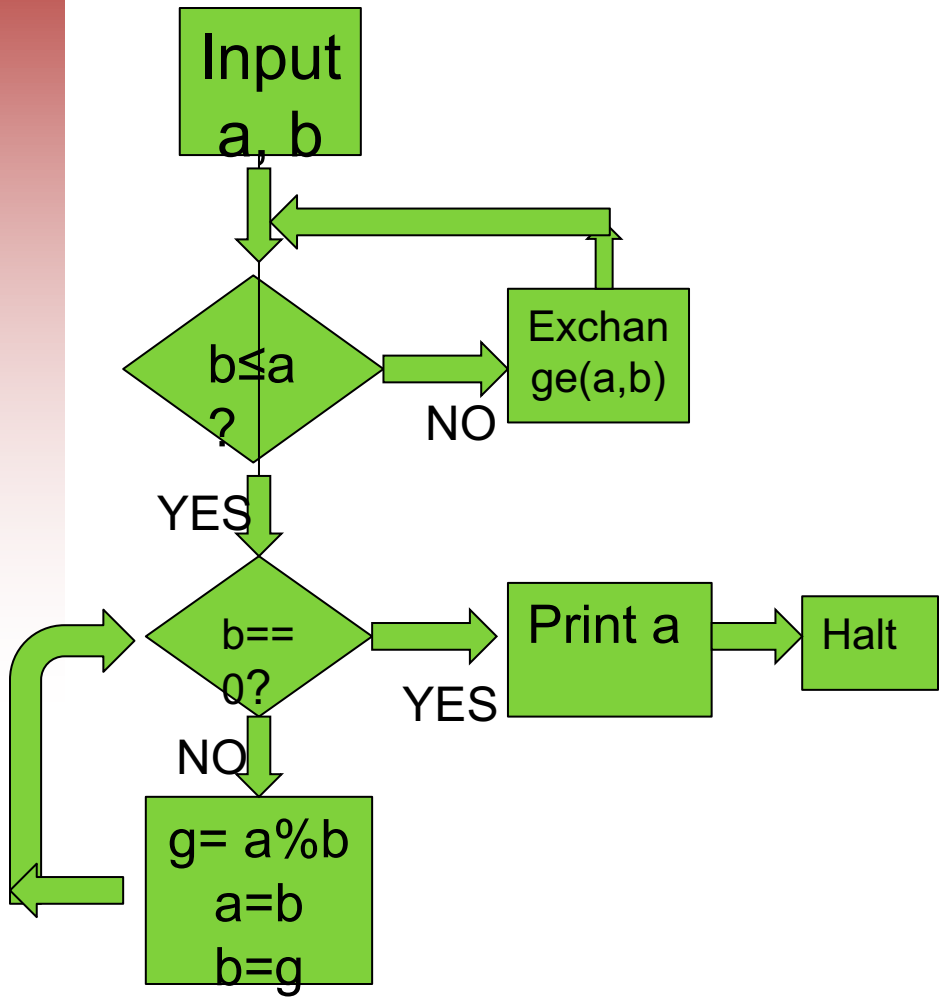
```
g = a%b;
a = b;
b = g;
```

After b = g

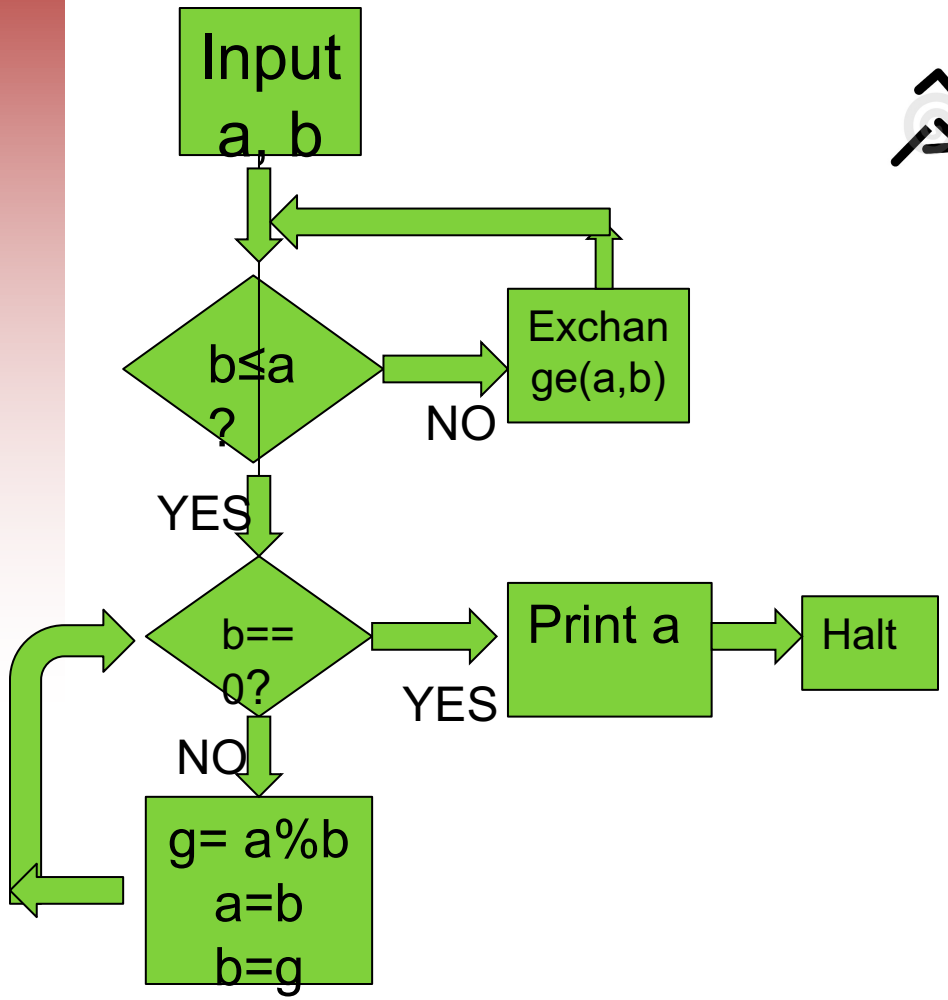


- Semi-colons give a sequential order in which to apply the statements.
- Variables are boxes to which a name is given.
- We have 3 variables: a, b, g. This gives us three boxes. Initially, a is 10, b is 6 and g is undefined.
- Run statements in sequence.
- Next statement to run

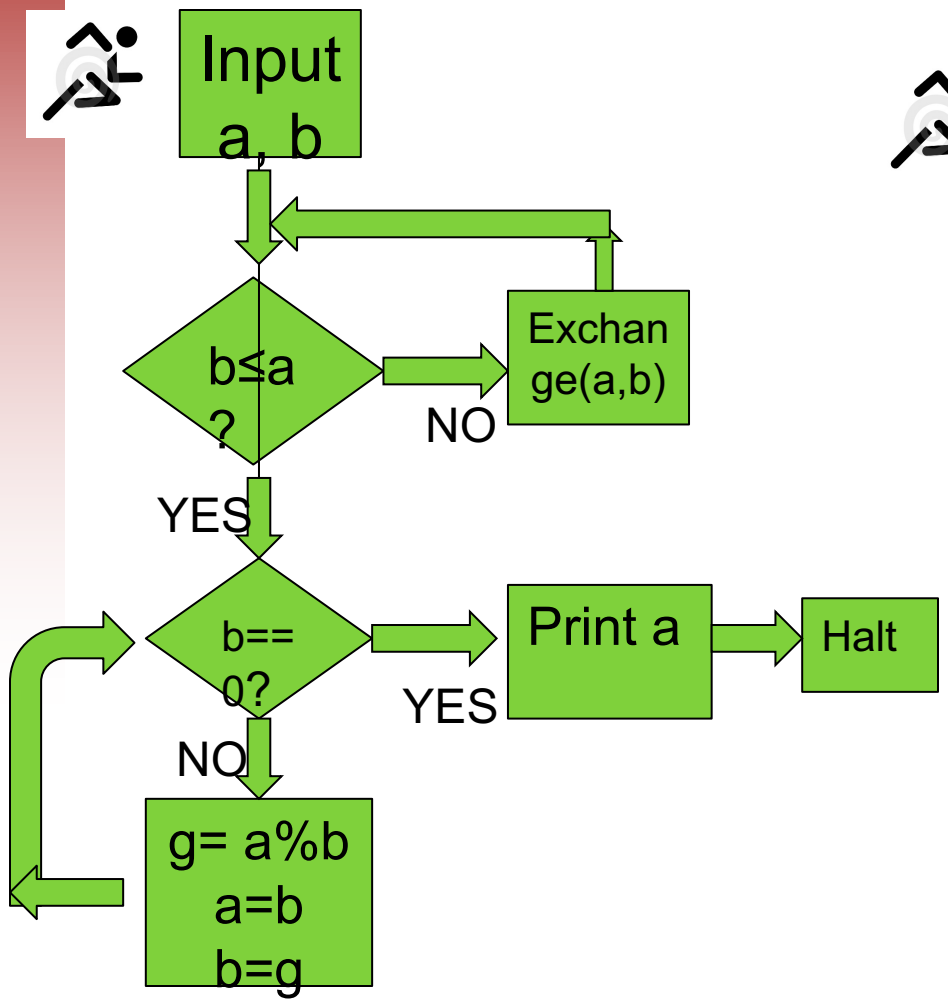
Running the program



Running the program

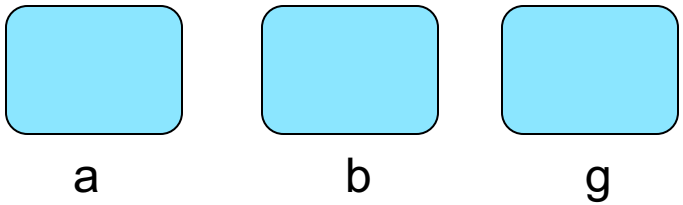


Running the program

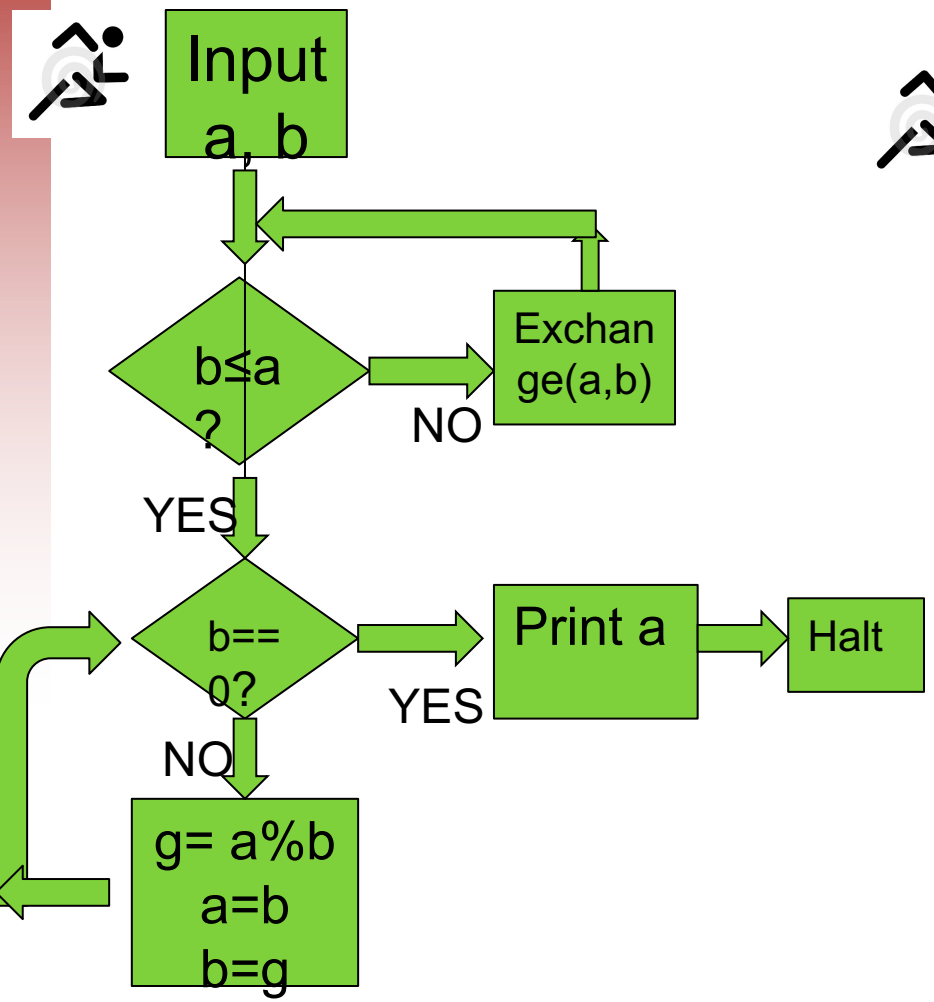


Program counter. At the next step to be executed. Initially at beginning.

State of the program is variables : boxes with names.

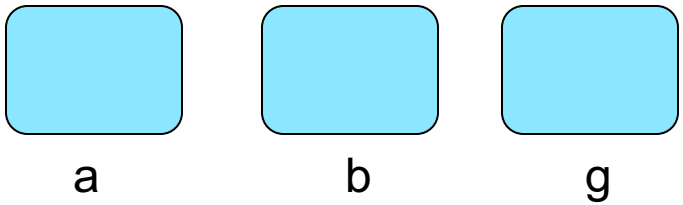


Running the program



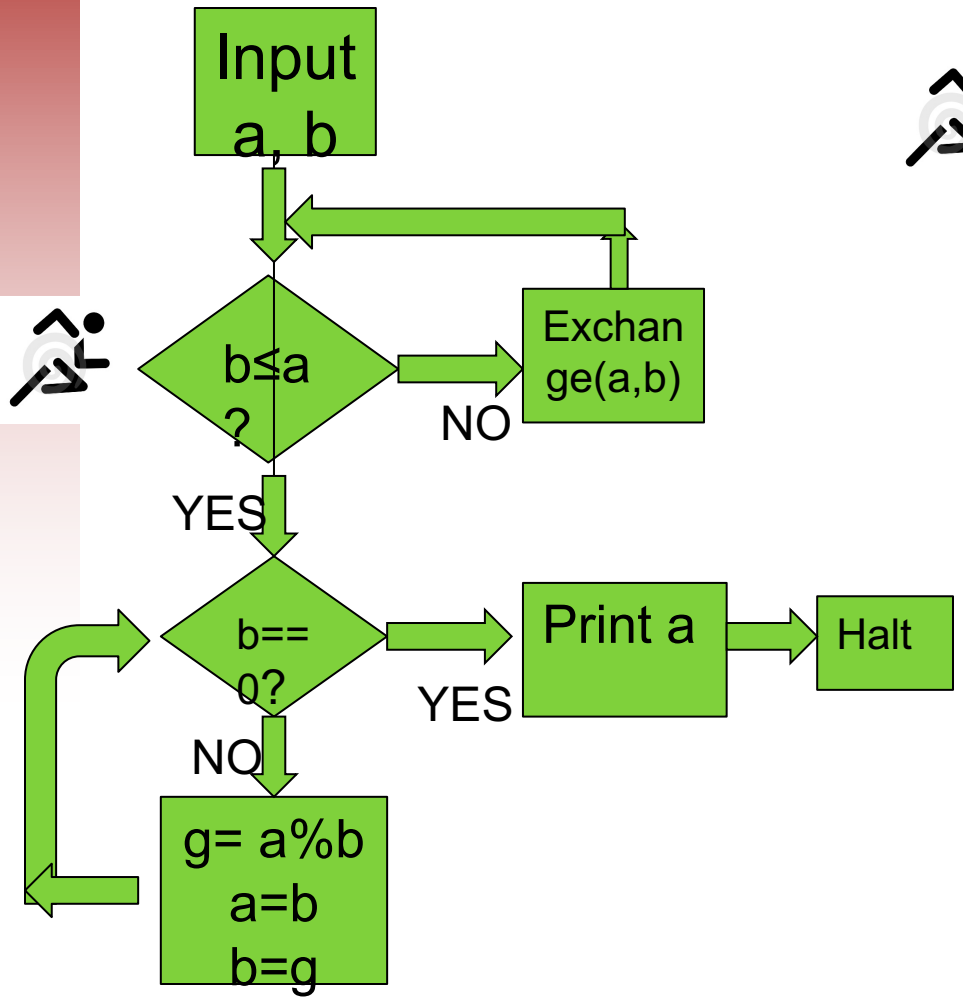
Program counter. At the next step to be executed. Initially at beginning.

State of the program is variables : boxes with names.



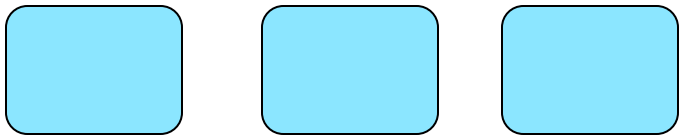
Now let us start running the flowchart. One step at a time.

Running the program



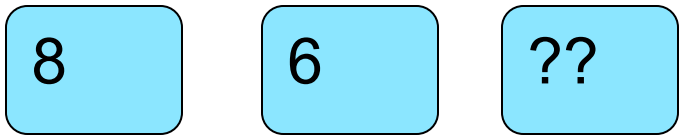
Program counter. At the next step to be executed. Initially at beginning.

State of the program is variables : boxes with names.

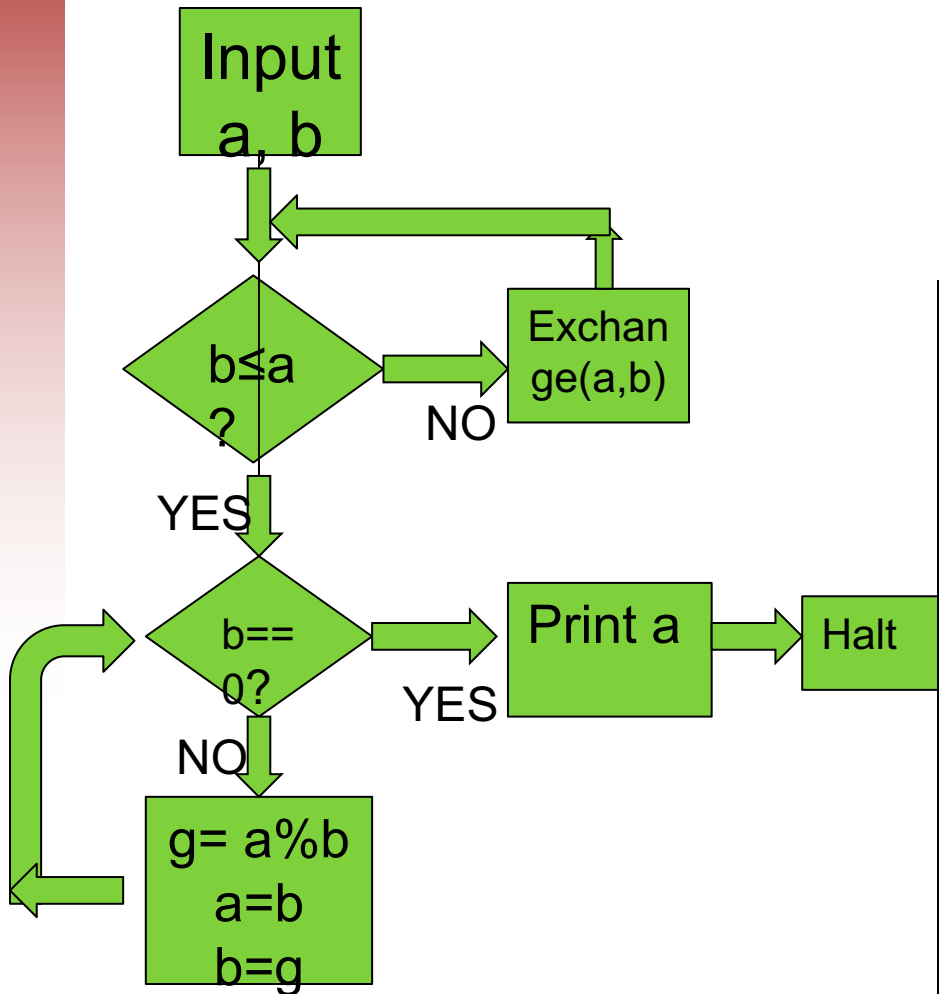


Now let us start running the flowchart. One step at a time.

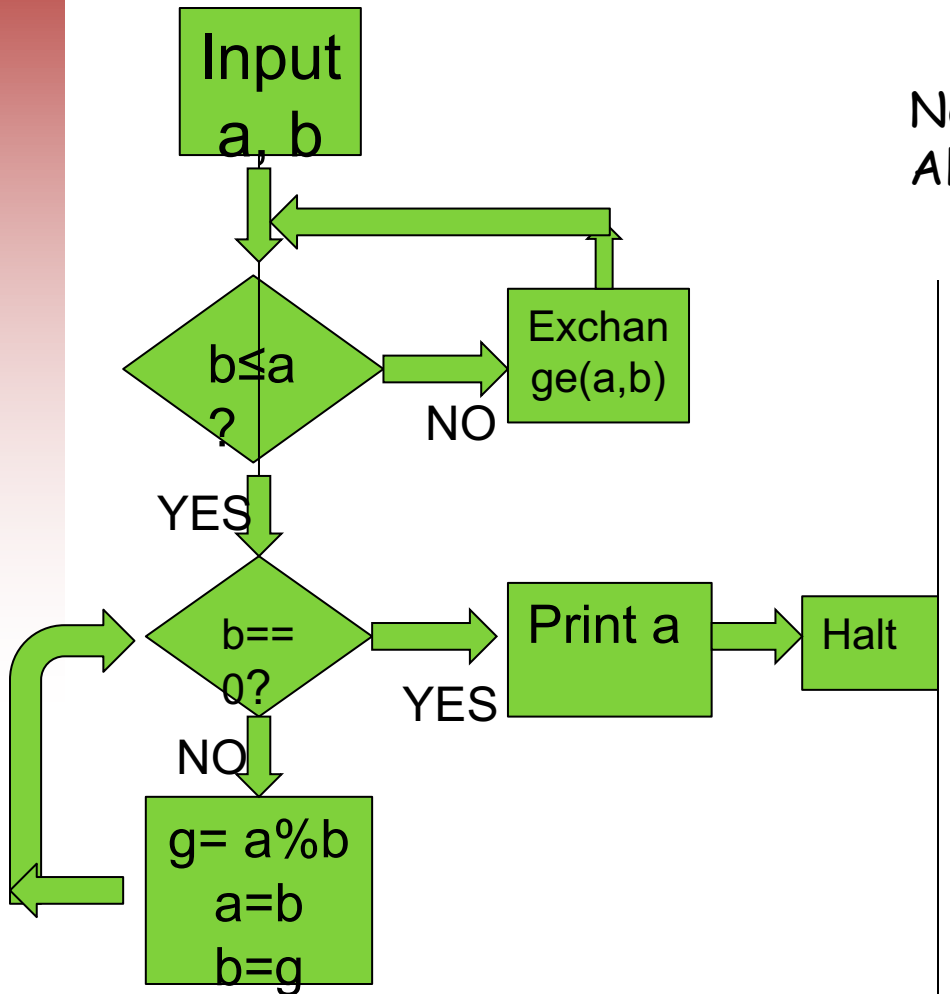
1. After input step:



Tracing the execution

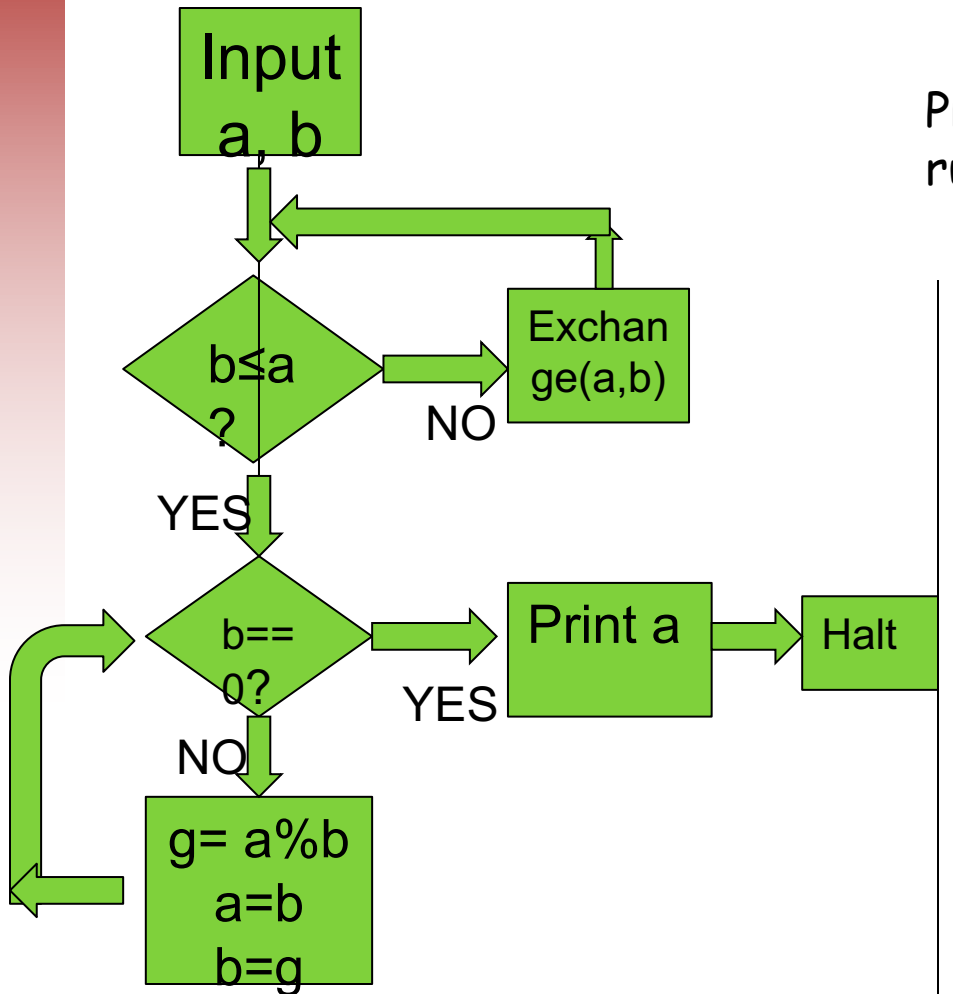


Tracing the execution



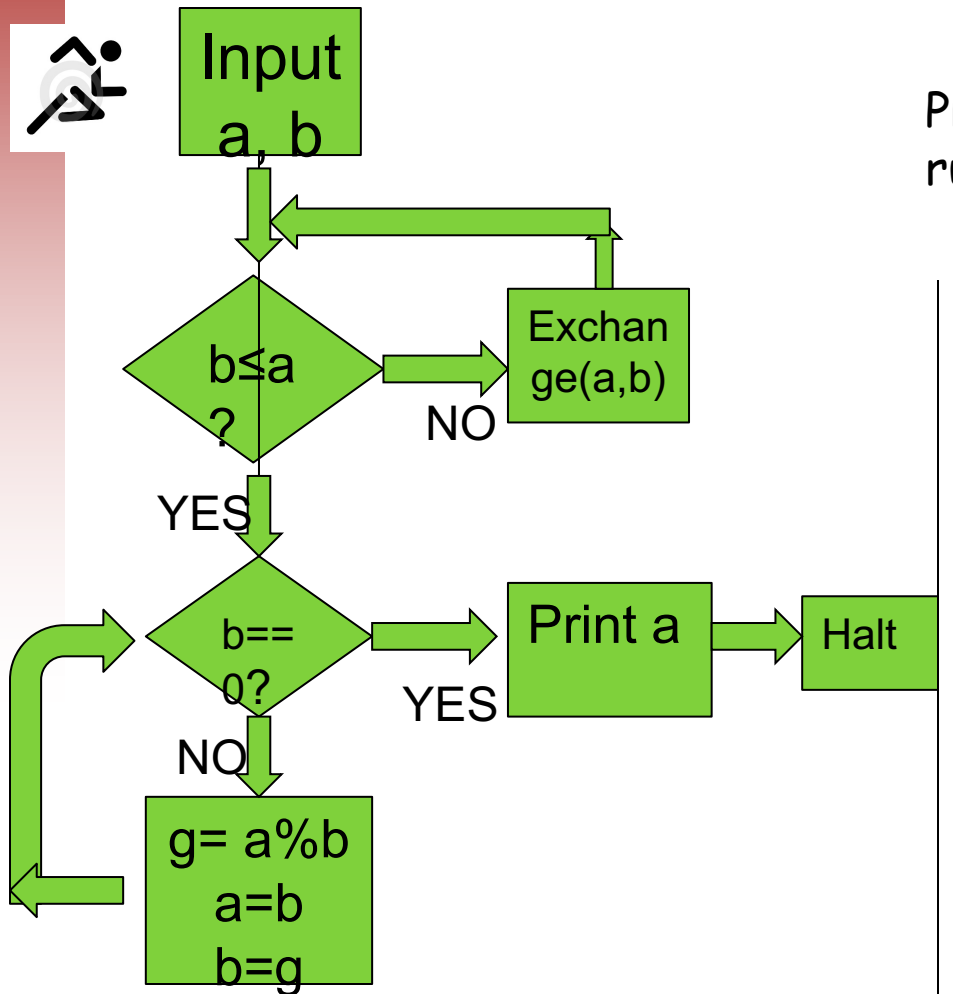
Now let us start running the flowchart.
Always one box at a time.

Tracing the execution



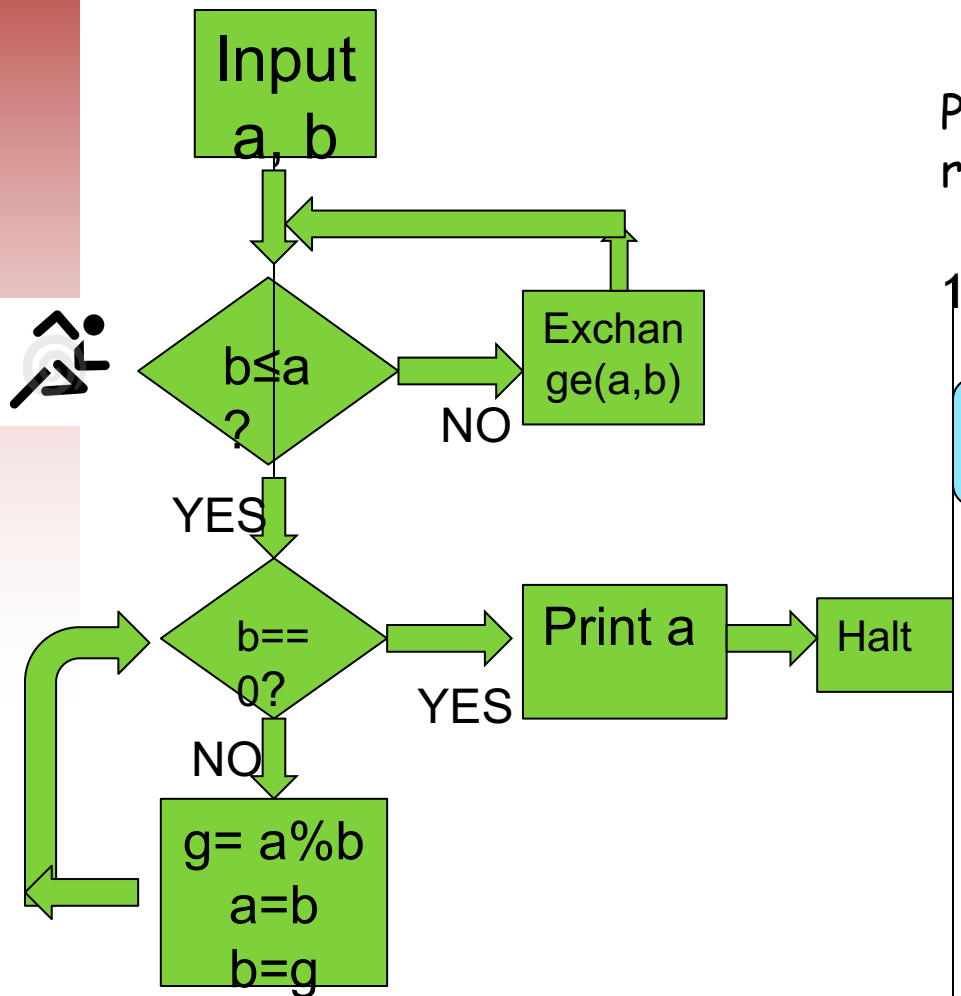
Program Counter is at the next step to be run

Tracing the execution



Program Counter is at the next step to be run

Tracing the execution

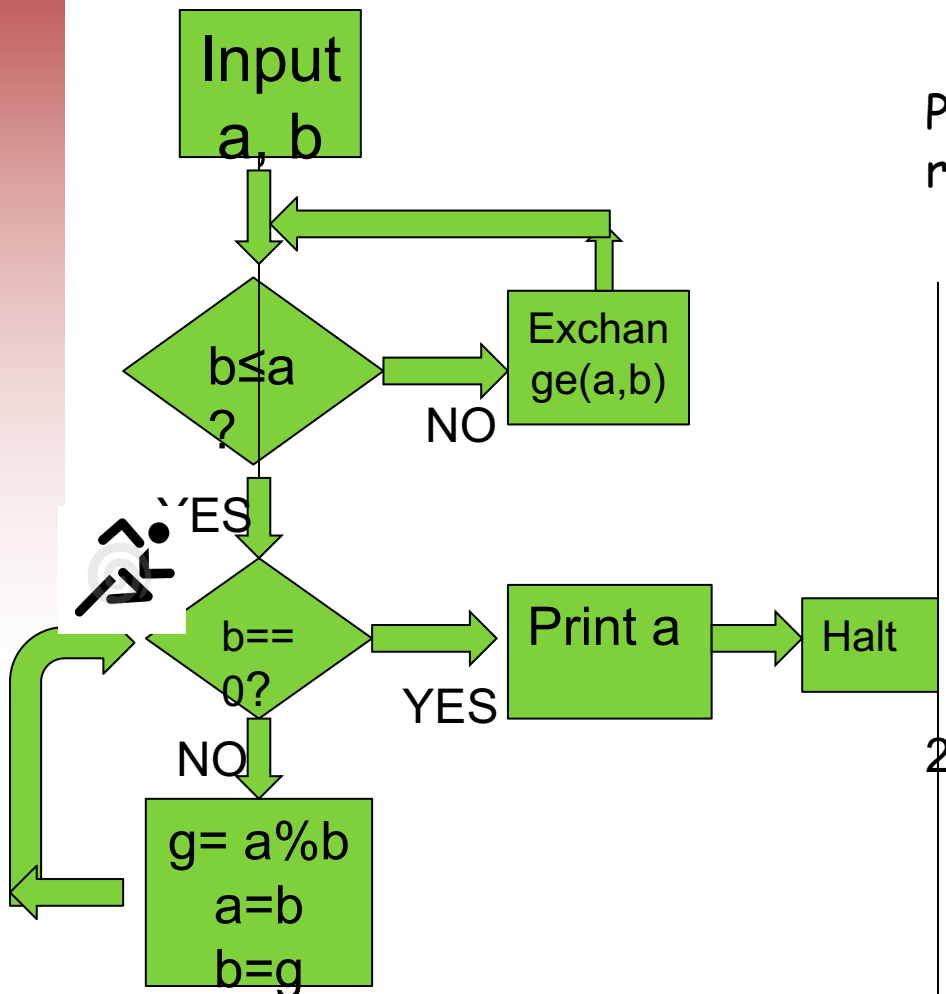


Program Counter is at the next step to be run

1. After input step:

8	6	??
a	b	g

Tracing the execution

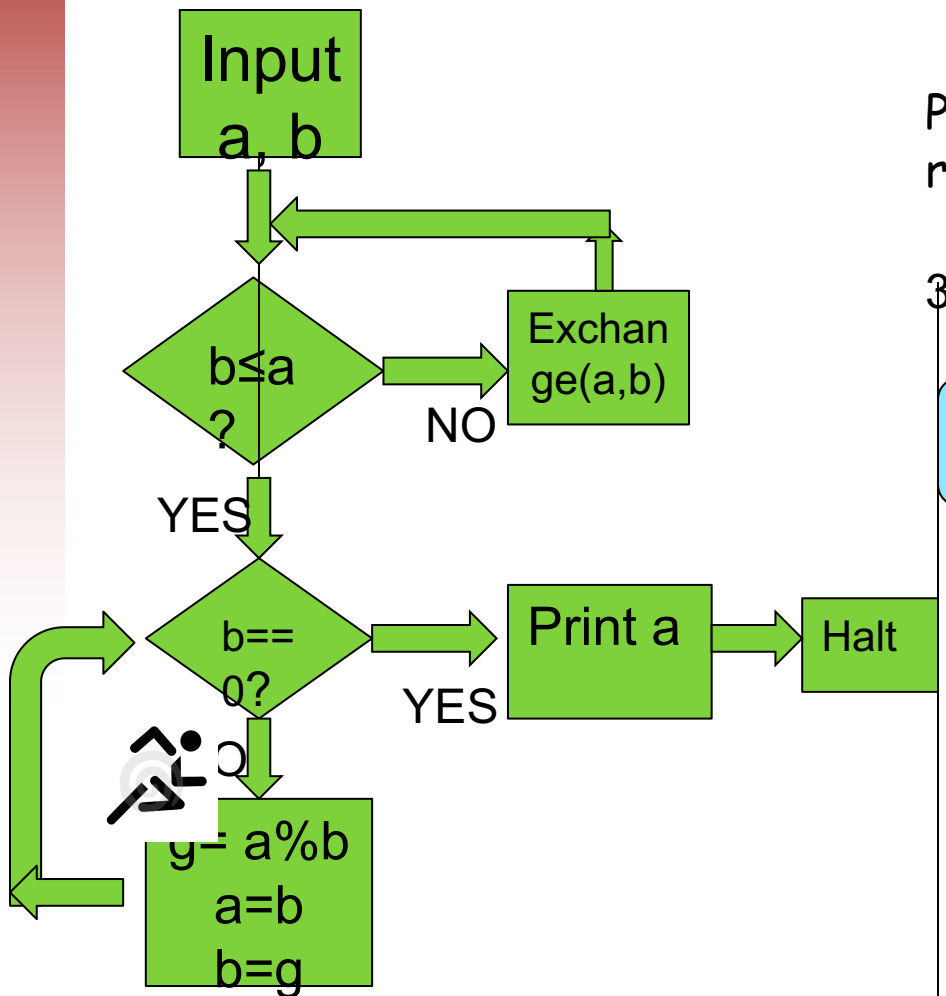


Program Counter is at the next step to be run

2. Test b < a? YES

8	6	??
a	b	g

Tracing the execution

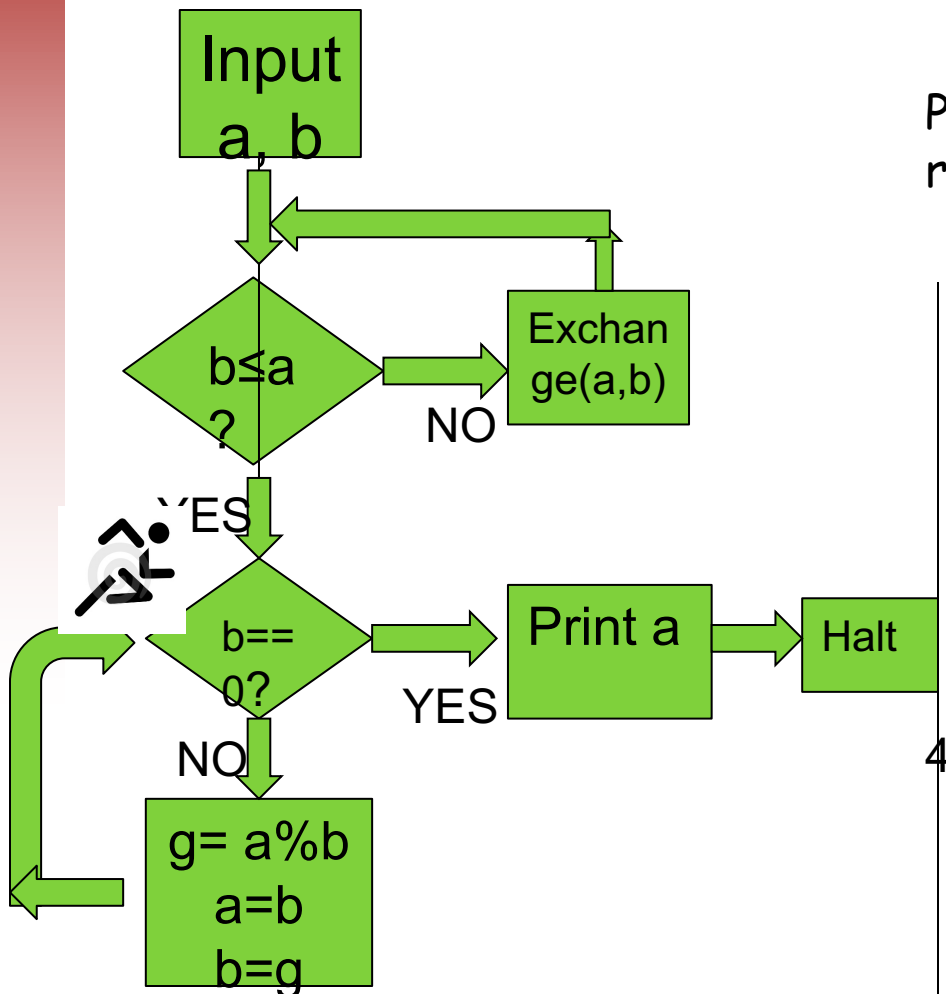


Program Counter is at the next step to be run

3. Test b==0? NO

a	b	g
8	6	??

Tracing the execution

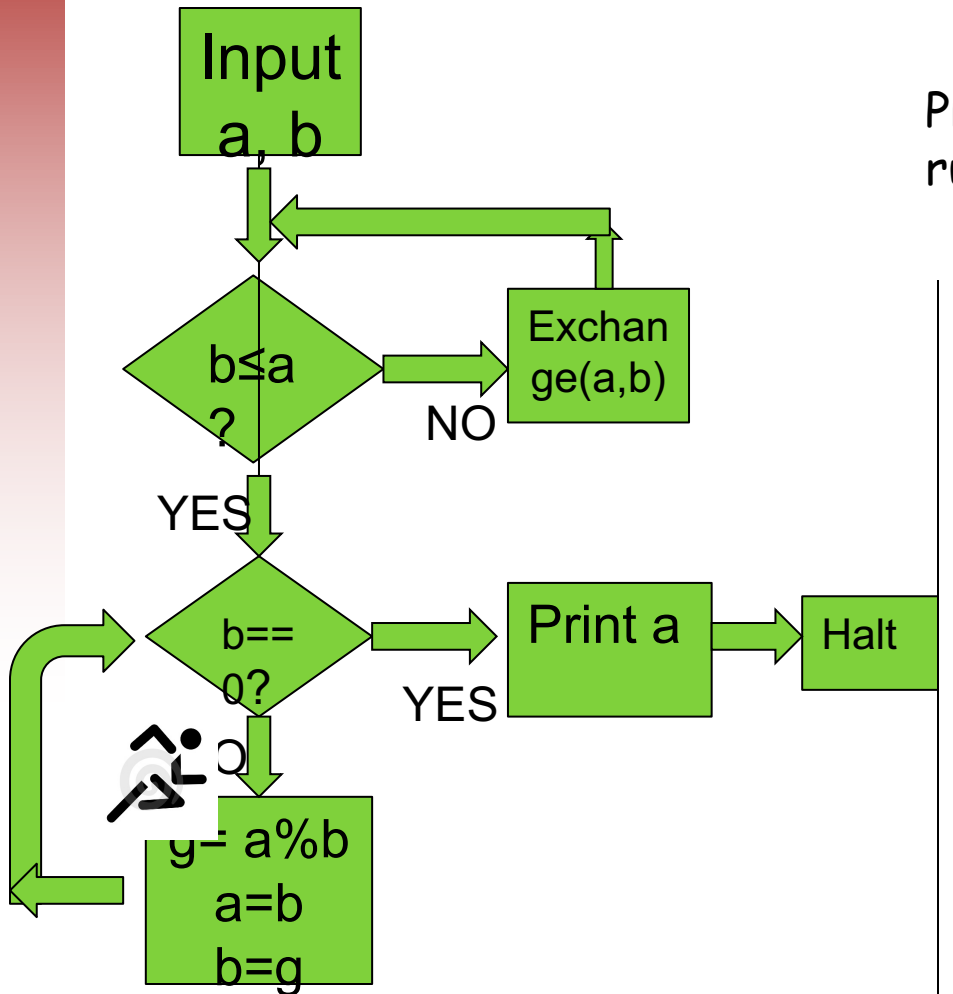


Program Counter is at the next step to be run

4. g=a%b; a=b; b=g;

6	2	2
a	b	g

Tracing the execution

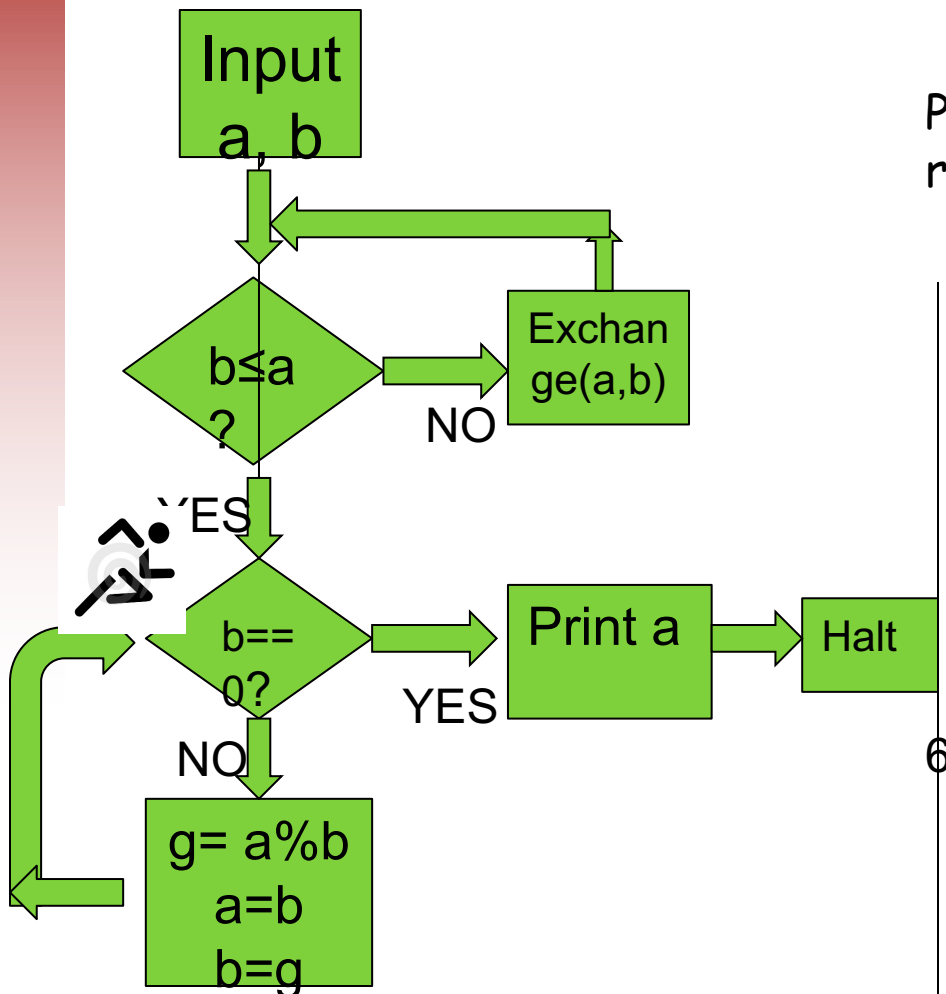


Program Counter is at the next step to be run

5. Test b==0? NO

a	b	g
6	2	2

Tracing the execution

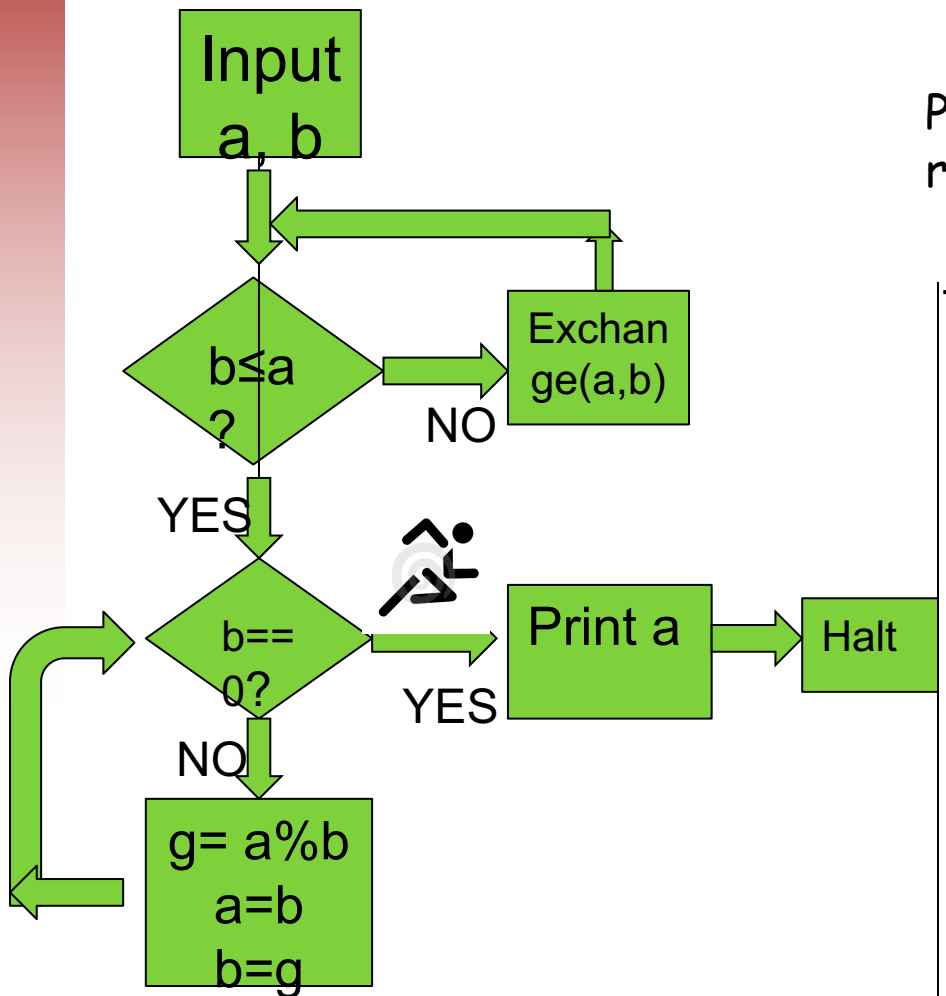


Program Counter is at the next step to be run

6. g=a%b; a=b; b=g;

a	b	g
2	0	0

Tracing the execution

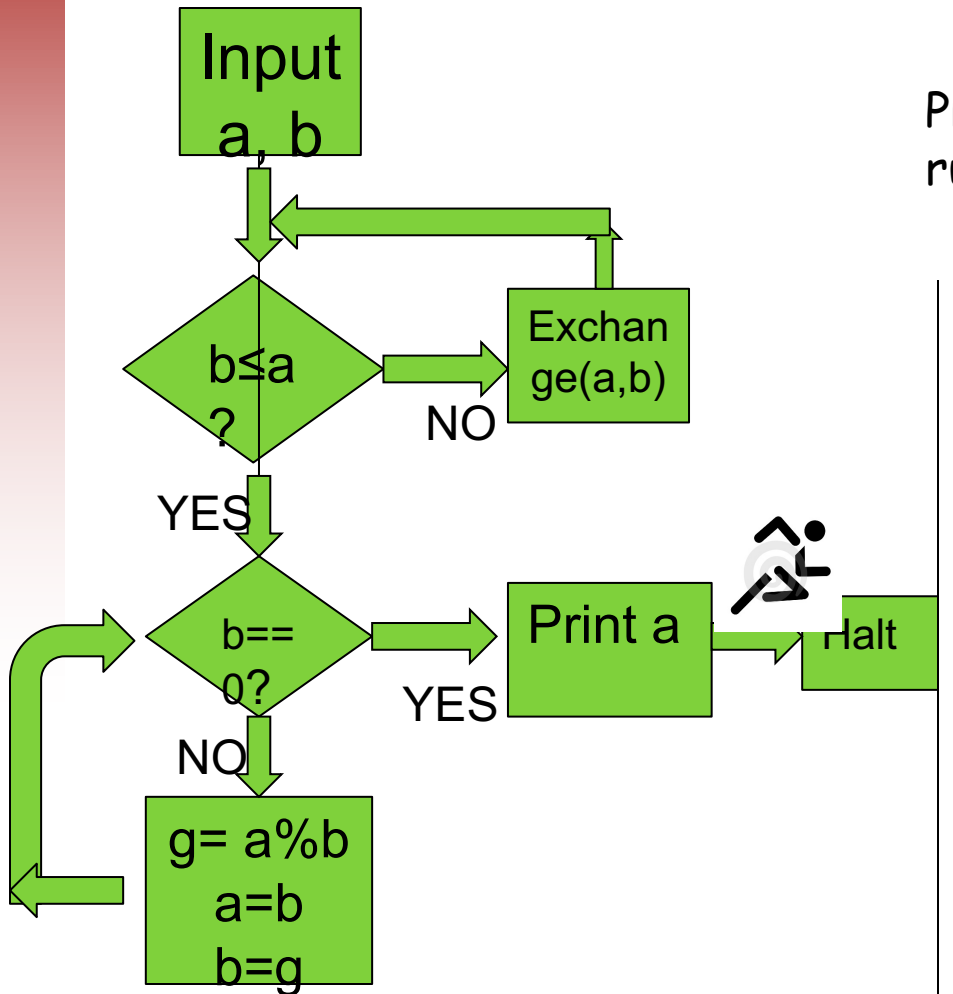


Program Counter is at the next step to be run

7. Test b==0? YES

a	b	g
2	0	0

Tracing the execution

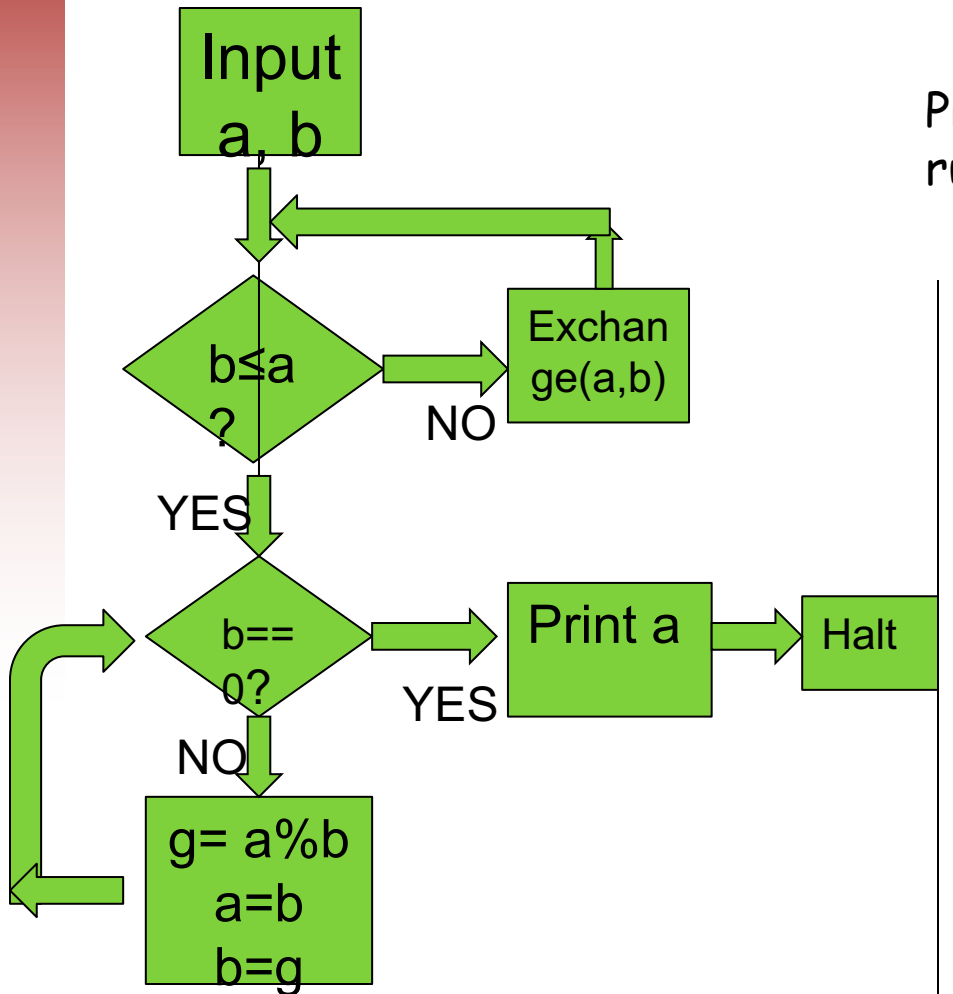


Program Counter is at the next step to be run

8. Print a

2

Tracing the execution



Program Counter is at the next step to be run

8. Print a

2

Multiple solutions and comparing them

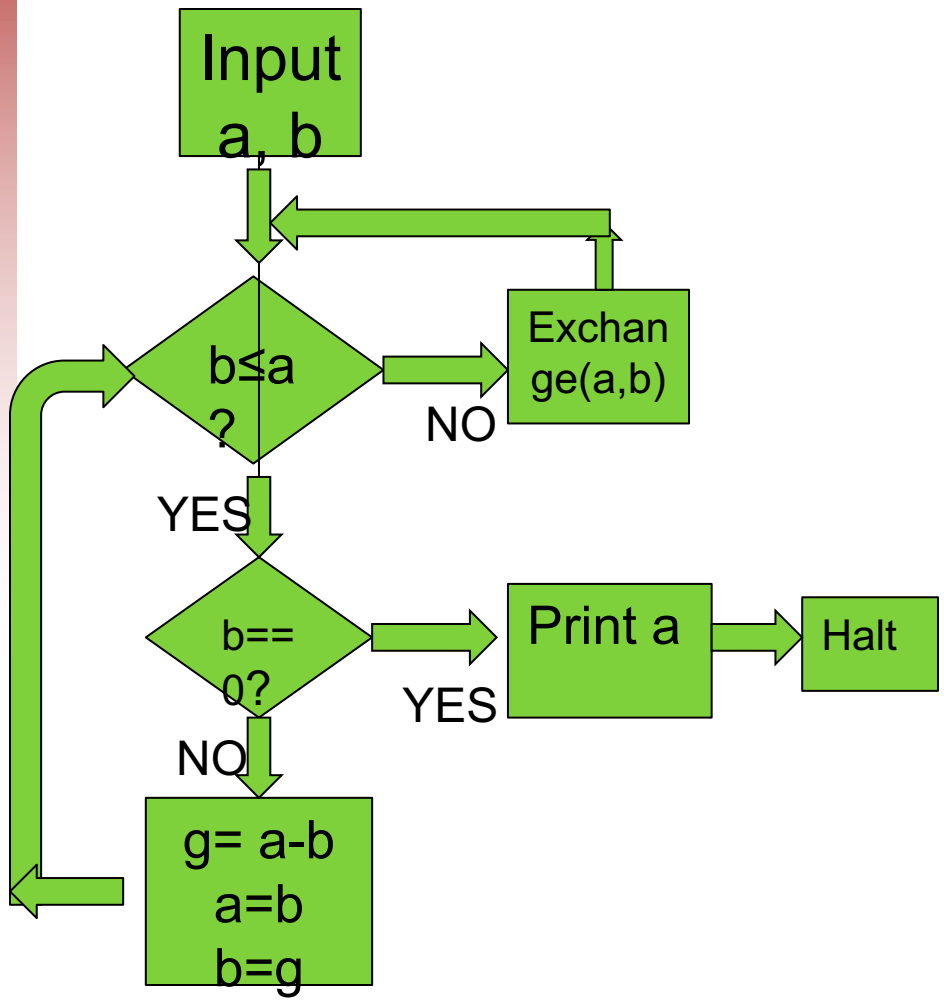
- How many times did we run in the loop? The fewer the better.

Multiple solutions and comparing them

- Multiple solutions are possible for the same problem.

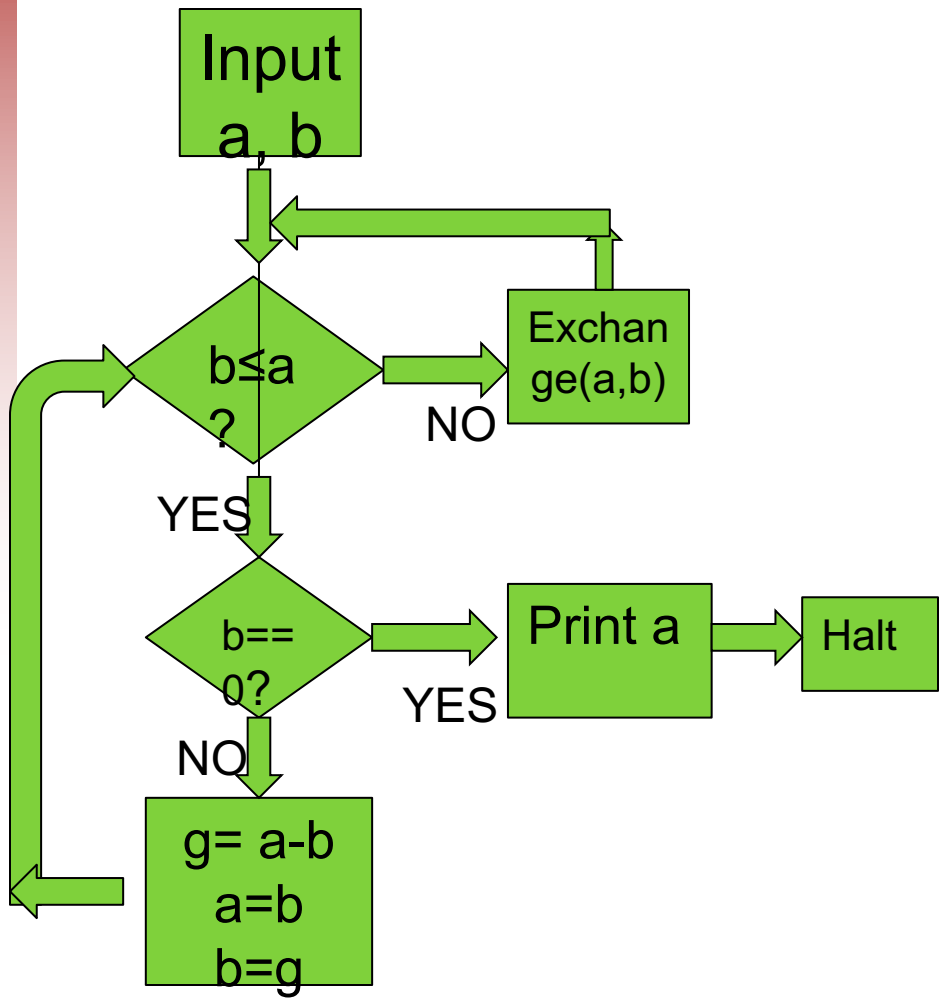
- How many times did we run in the loop? The fewer the better.

Multiple solutions and comparing them



- Multiple solutions are possible for the same problem.
- Is the adjacent flowchart correct for gcd?
- How many times did we run in the loop? The fewer the better.

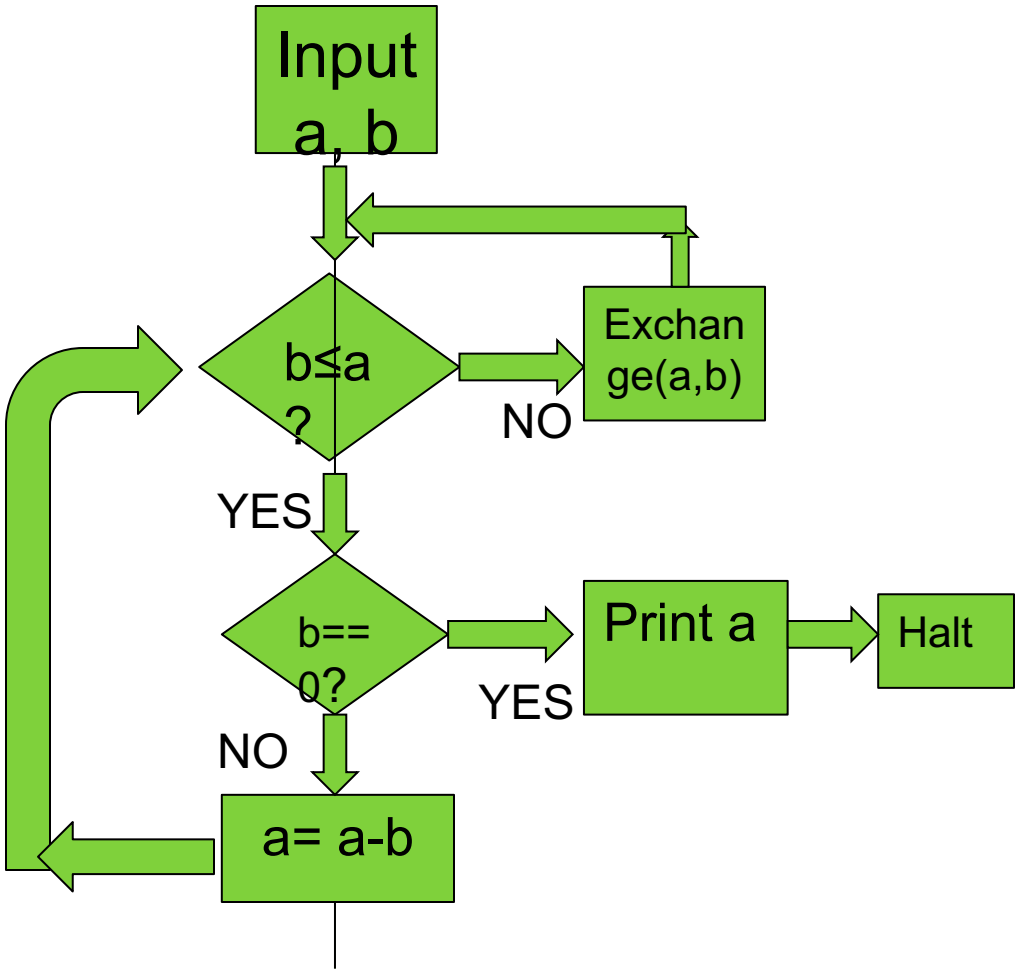
Multiple solutions and comparing them



- Multiple solutions are possible for the same problem.
- Is the adjacent flowchart correct for gcd?
- How many times did we run in the loop? The fewer the better.
- Is it seriously more: by an order of magnitude? Notion of complexity.

Another solution

- A (slower) alternative. How many times does the loop iterate?



Acknowledgments: This lecture slide is based on the material prepared by Prof. Sumit Ganguly, CSE, IIT Kanpur. The slide design is based on a template by Prof. Krithika Venkataramani.