ESC101: Introduction to Computing

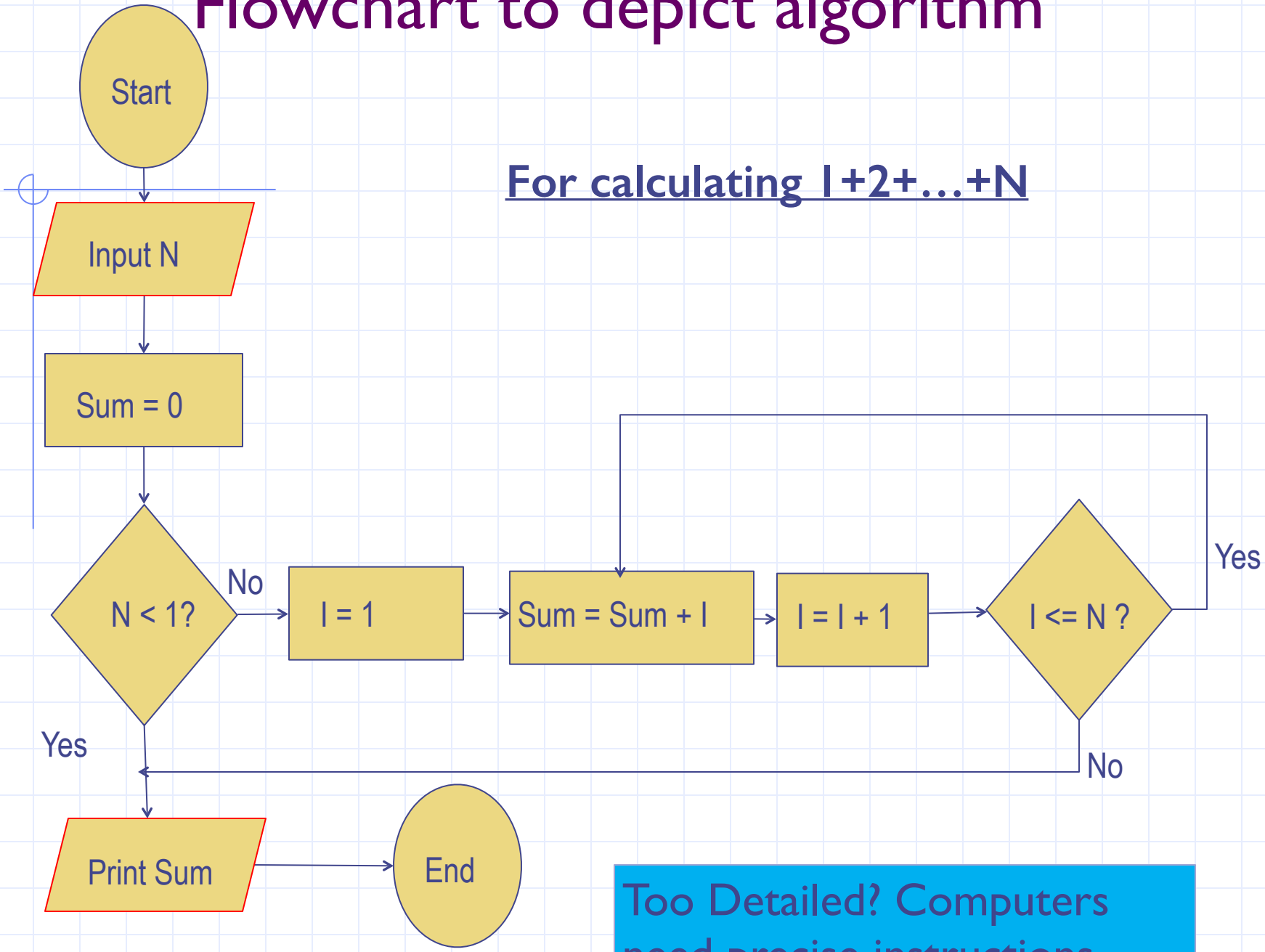
Overview of Programming

Process of Programming

- Obtain a logical solution to your problem.
- A logical solution is a finite and clear step-by-step procedure to solve your problem.
- Also called an Algorithm (or *recipe*).
  - We can visualize this using a Flowchart.
  - Very important step in the programming process.

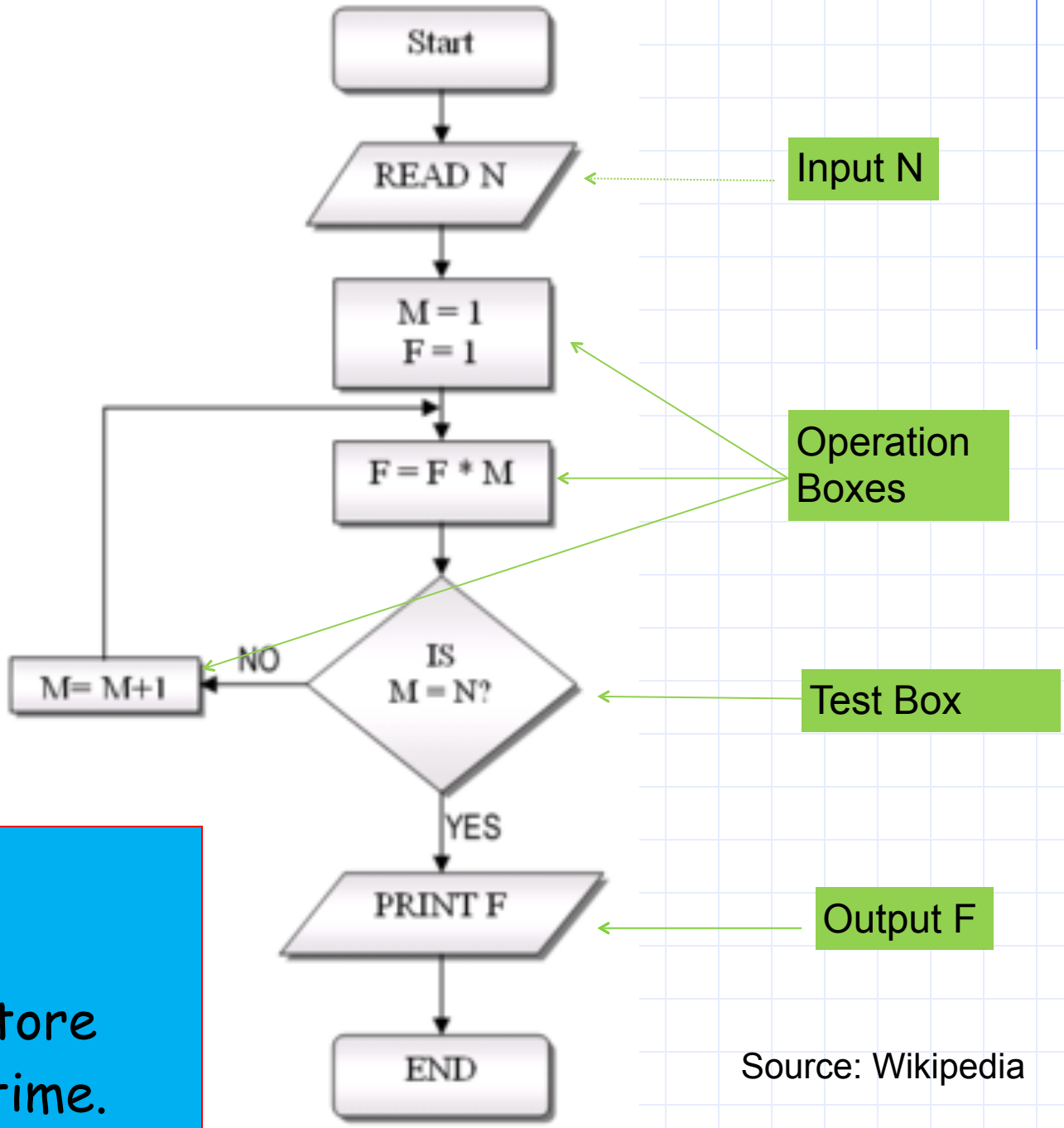
Flowchart to depict algorithm

For calculating 1+2+…+N



Too Detailed? Computers need precise instructions

Flowchart to calculate the factorial of N



Source: Wikipedia

Notation:

M,F,N are called variables. They store one number at a time.

Greatest Common Divisor

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

Greatest Common Divisor

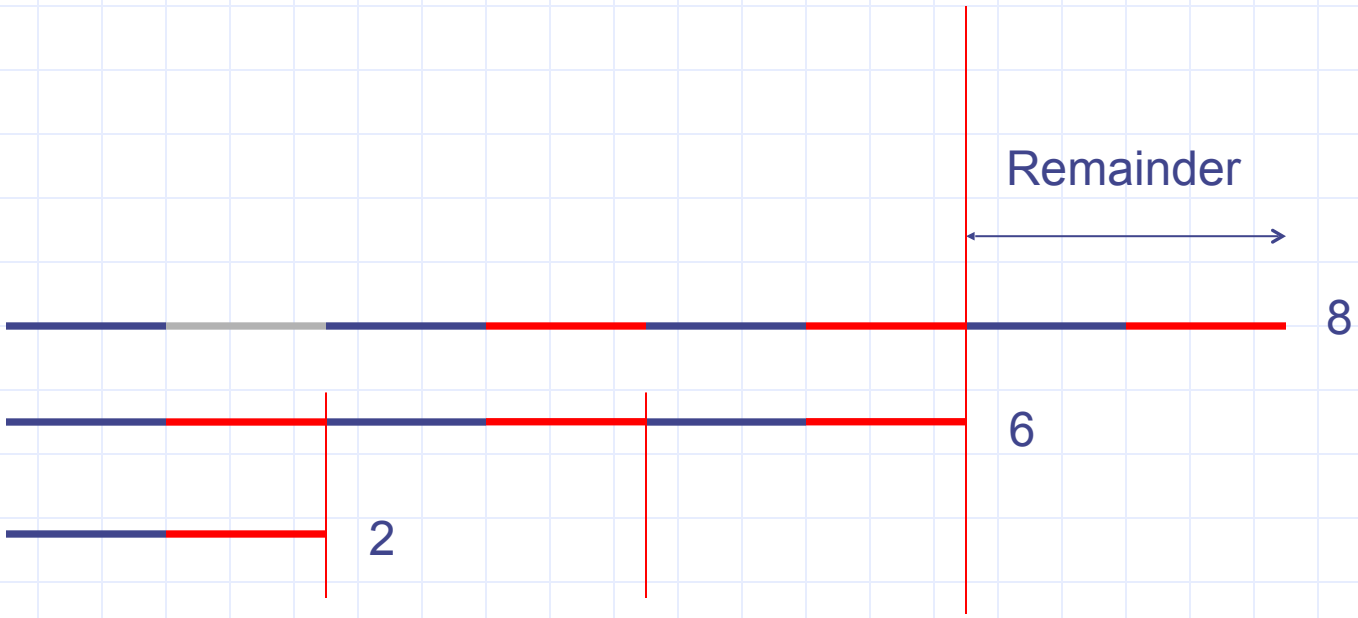
- This will compute gcd correctly, but is VERY slow (think about large numbers m and n=m-1).
- Can it be done faster?

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



Gcd(8, 6) = 2.

GCD Algorithm - Intuition

102

102 mod 21 = 18

21

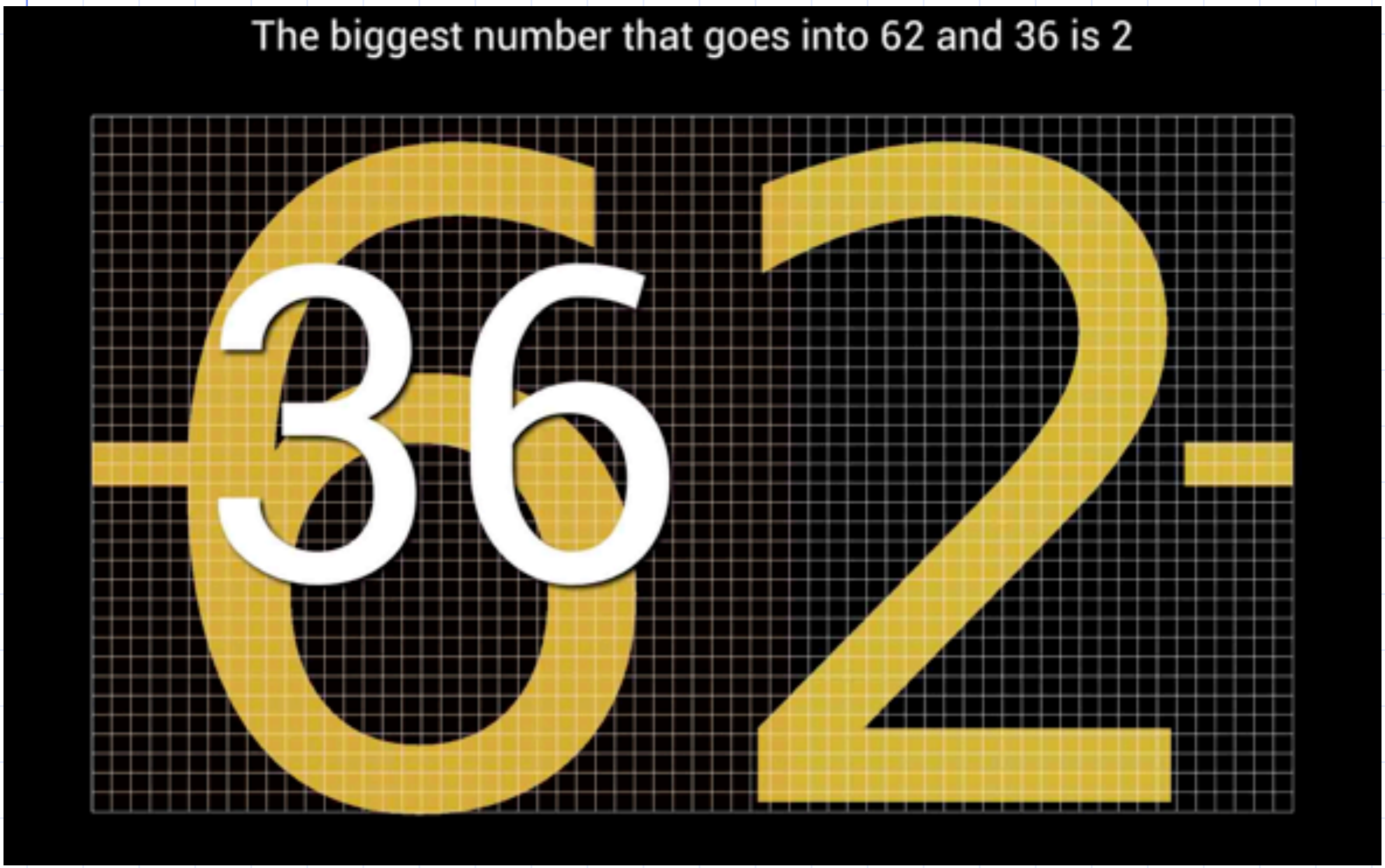
21 mod 18 = 3

18

3

Gcd (102, 21) = 3

GCD Algorithm - Intuition



source: wikipedia

Euclid's method for gcd

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

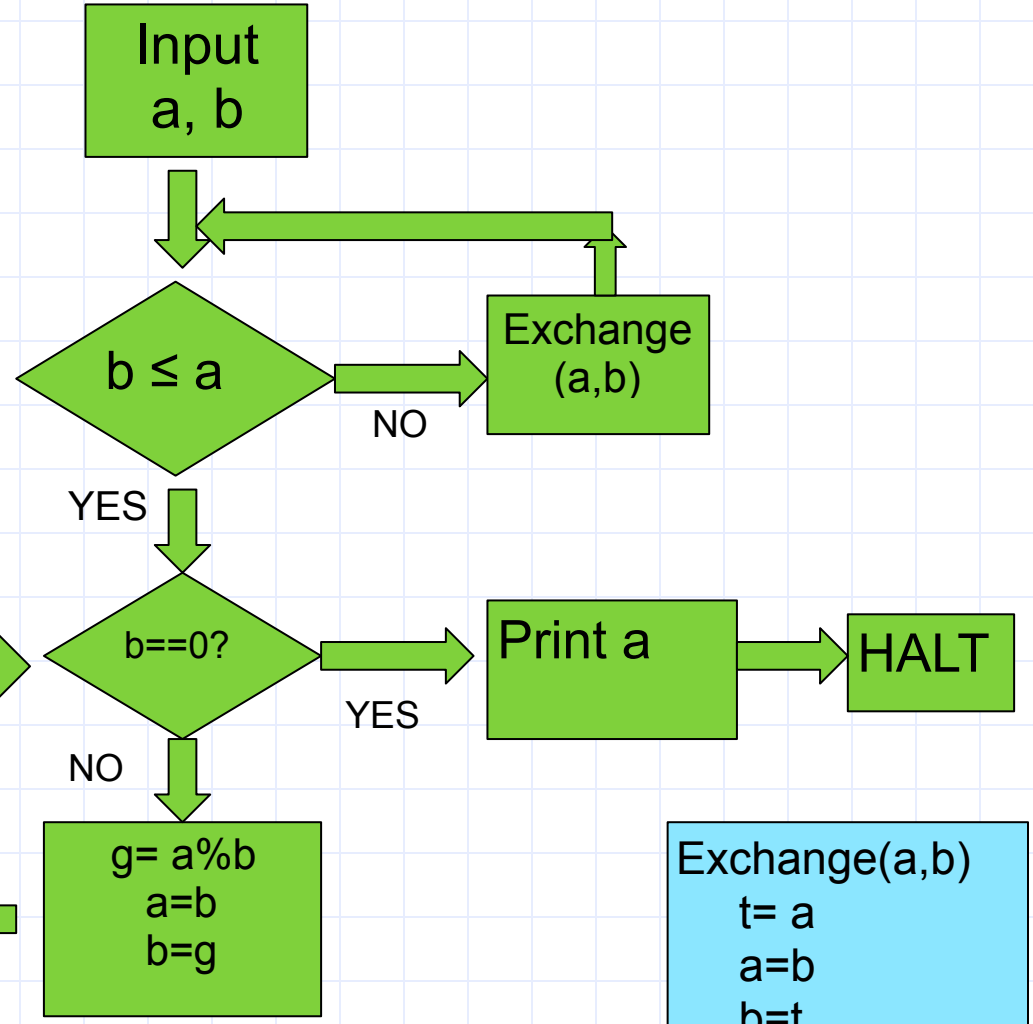
Suppose a > b. Then the gcd of a and b is the same as the gcd of b and the remainder of a when divided by b.

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

To see this consider division of a by b

$a = bq + r$

Euclid's gcd

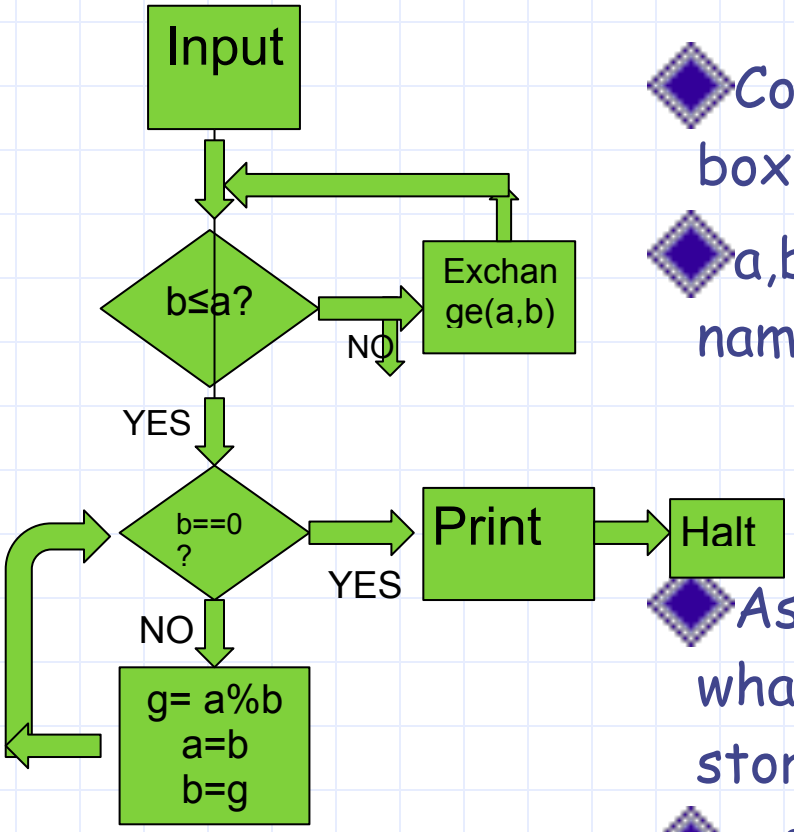


a,b,g are variables. Variables "store" exactly one value at a time.

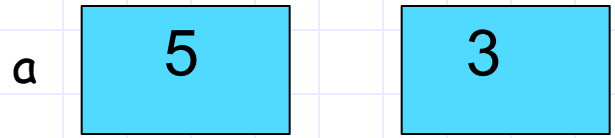
a%b is the remainder when a is divided by b.
Eg. 8%3 is 2

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

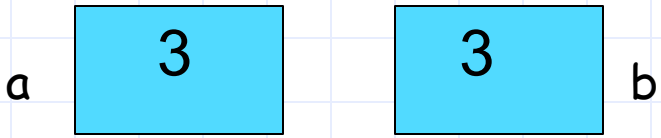
Variables and Assigning them



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



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



Sequential assignments

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

initially

a	b	g
10	6	??

- 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

a	b	g
10	6	4

- 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

a	b	g
6	6	4

- 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

a	b	g
6	4	4

- 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

GCD Algorithm

Data: Integers m and n

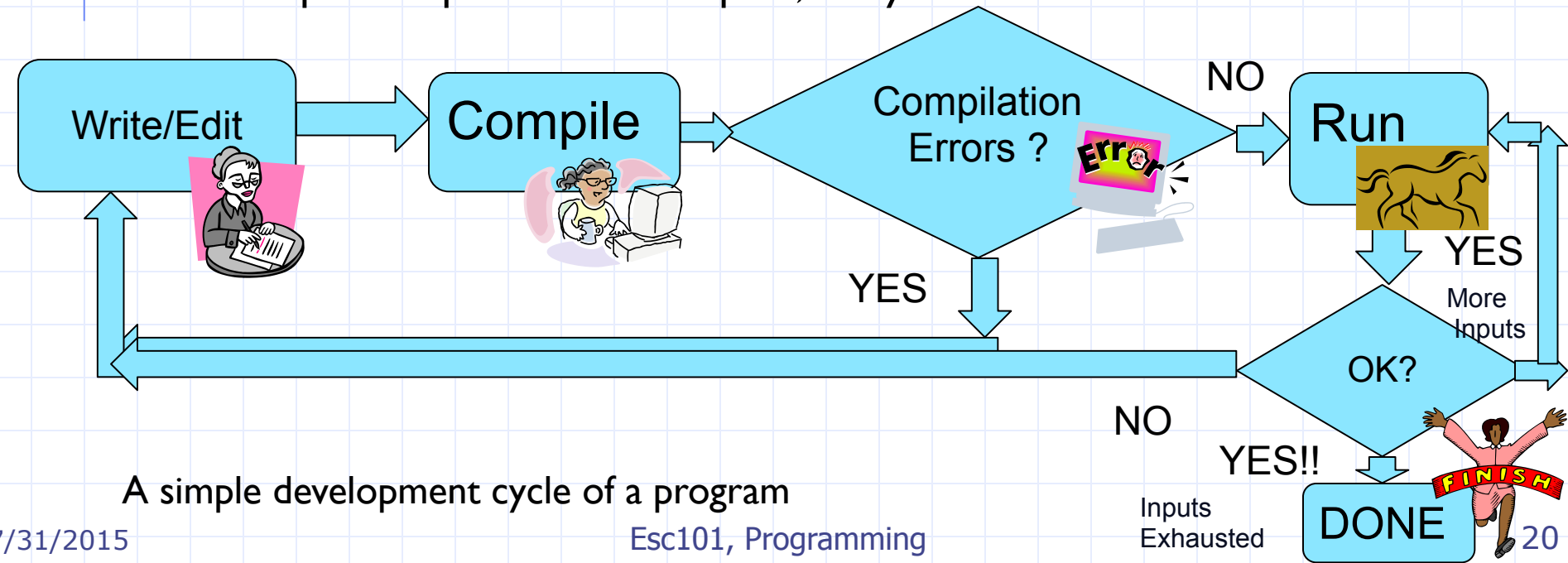
```
If n > m then interchange m and n;
        g ← m %n;
        m ← n;
        n ← g;
end
return m;
```

Overview of Programming

Using C

The Programming Cycle

1. Write your program or edit (i.e., change or modify) your program.
2. Compile your program. If compilation fails, return to editing step.
3. Run your program on an input. If output is not correct, return to editing step.
   a. Repeat step 3 for other inputs, if any.



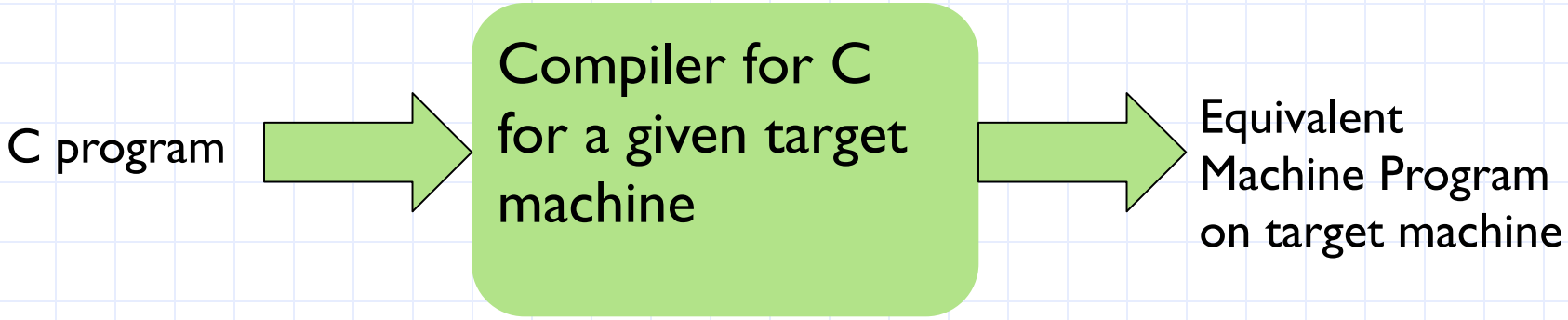
A simple development cycle of a program

IDE for Edit-Compile-Run cycle

- In this course, you will be using an Integrated Development Environment (IDE). IDE will be available through your browser.
- First login to the system.
- Type in your program in the editor of the IDE.
- Use the compile button to compile.
- Run button to run.
  - The labs in the first week will introduce you to the system in more detail.

Why program in high level languages like C

- Writing programs in machine language is long, tedious and error-prone.
- They are also not portable—meaning program written for one machine may not work on another machine.
- Compilers work as a bridge.
- Take as input a C program and produce an equivalent machine program.



A Simple Program

- **Today we will see some simple C programs.**

```
# include <stdio.h>
int main ()   {
    printf("Welcome to ESC101");
    return 0;
}
```

The program prints the message "Welcome to ESC101"

Program components

```
# include <stdio.h>

int main ()
{
    printf("Welcome to ESC101");
    return 0;
}
```

1. This tells the C compiler to include the standard input output library.

2. Include this line routinely as the first line of your C file.

printf is the function called to output from a C program. To print a string, enclose it in " " and it gets printed. For now, do not try to print " itself.

"return" returns the control to the caller (program finishes in this case.)

main() is a function. All C programs start by executing from the first statement of the main function.

printf("Welcome to ESC101"); is a statement in C. Statements in C end in semicolon ;

printf

- printf is the "voice" of the C program
  - Used to interact with the users
- printf prints its arguments in a certain format
  - Format provided by user

Modified Simple Program

```
# include <stdio.h>
int main ()   {
   printf("Hello! \n Welcome to ESC101");
   return 0;
}
```

The program prints the message

Hello!
Welcome to ESC101

Another Simple Program

- **Program to add two integers (17 and 23).**

```
# include <stdio.h>
int main ()  {
    int a = 17;
    int b = 23;
    int c;
    c = a + b;
    printf("Result is %d", c);
    return 0;
}
```

The program prints the message: **Result is 40**

```
# include <stdio.h>
int main ()
{
  int a = 17;
  int b = 23;
  int c;
  c = a + b;

  printf("Result is %d",c);
  return 0;
}
```

This tells the compiler to reserve a "box" large enough to hold an integer value. The box is named "a" for use in the rest of the program.

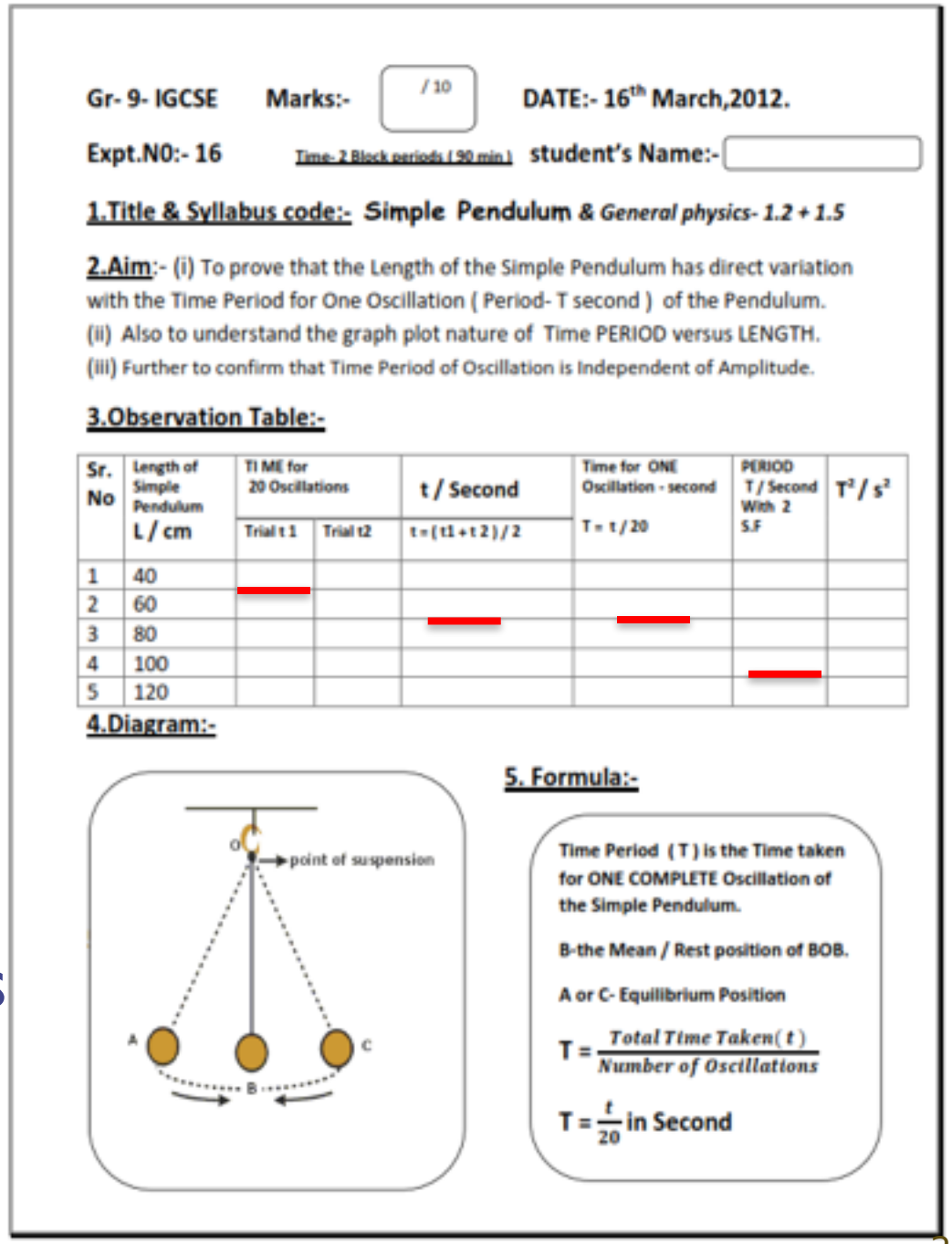
"= 17" stores value 17 in the box that we have named "a".
It is OK to skip this part and store value later as we do for box named "c".

+ is an operator used to add two numbers. The numbers come from the values stored in the boxes

%d tells printf to expect one integer argument whose value is to be printed. We call it placeholder.
We will see more

printf (% format)

- % format specifiers allow C program to print things whose values are yet not computed
  - will be known while running the program
- %... is similar to the blanks in a lab sheets used for phy/chem labs



Gr- 9- IGCSE Marks:- / 10 DATE:- 16th March,2012.

Expt.N0:- 16 Time- 2 Block periods (90 min) student's Name:-

1.Title & Syllabus code:- **Simple Pendulum** & *General physics- 1.2 + 1.5*

2.Aim:- (i) To prove that the Length of the Simple Pendulum has direct variation with the Time Period for One Oscillation (Period- T second) of the Pendulum.
(ii) Also to understand the graph plot nature of Time PERIOD versus LENGTH.
(iii) Further to confirm that Time Period of Oscillation is Independent of Amplitude.

3.Observation Table:-

Sr. No	Length of Simple Pendulum L / cm	TI ME for 20 Oscillations		t / Second	Time for ONE Oscillation - second	PERIOD T / Second With 2 S.F	T^2 / s^2
		Trial t 1	Trial t2	t = (t1 + t 2) / 2	T = t / 20		
1	40						
2	60						
3	80						
4	100						
5	120						

4.Diagram:-

5. Formula:-

Time Period (T) is the Time taken for ONE COMPLETE Oscillation of the Simple Pendulum.

B-the Mean / Rest position of BOB.

A or C- Equilibrium Position

$$T = \frac{Total\ Time\ Taken(t)}{Number\ of\ Oscillations}$$

$$T = \frac{t}{20} \text{ in Second}$$

Another Simple Program

- **A smaller program to add two integers (17 and 23).**

```
# include <stdio.h>
int main ()  {
  printf("Result is %d", 17+23);
  return 0;
}
```

The program prints the message "Result is 40"

In this case + is operating directly on two integer **constants**.

Next class

- Types