FACULTY OF COMPUTING
& INFORMATION TECHNOLOGY
KING ABDULAZIZ UNIVERSITY

FCIT
KAU

كلية الحاسبات
وتقنية المعلومات
جامعة الملك عبدالعزيز

Chapter 0
Introduction to Problem-Solving

CPIT 110 (Problem-Solving and Programming)

Version 2.0

Sections

- 0.1. Problem-Solving & Computer Science
- 0.2. Program Design & Problem-Solving Techniques
- 0.3. Steps in Program Development
- 0.4. Algorithms, Pseudocode, & Flowcharts
- 0.5. Decision Structures

Python Online IDE

Examples

- Example 1: Road Example
- Example 2: Area of a Rectangle Calculator
- Example 3: Simple Calculator
- Example 4: Determining a Student's Final Grade
- Example 5: Converting The Length
- Example 6: Area of a Rectangle Calculator
- Example 7: Determining The Largest Value

Objectives

- To explain what problem solving is, and why it is important (0.1).
- To understand how to write algorithms (0.1–0.5).
- To describe how a program can be designed (0.2–0.3).
- To describe algorithms in different forms (0.4).
- To understand the difference between algorithms and pseudocode (0.4).
- To draw program flowcharts (0.4-0.5).
- To understand decision Structures (0.5).

0.1. Problem-Solving & Computer Science

- What is Computer Science?
- Example 1: Road Example
- Algorithms
- Example 2: Area of a Rectangle Calculator

What is Computer Science?

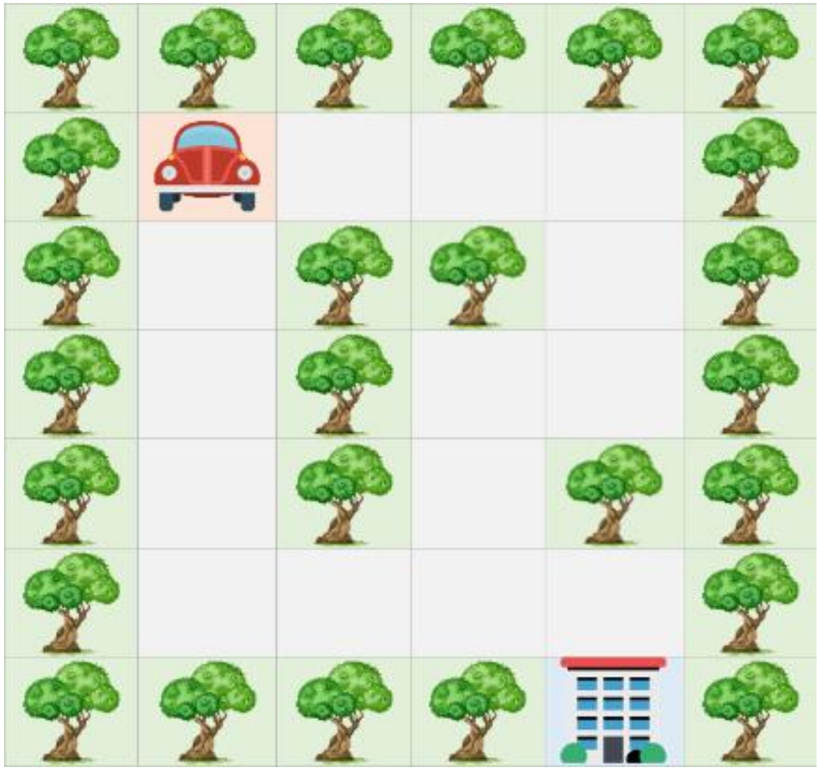
- Computer Science can be summarized with two simple words: problem solving.
- Computer Science is the study of problems, problem-solving, and the solutions that come out of this problem-solving process.
- Given a problem, the goal is to develop an algorithm to solve the problem.
- An algorithm is a step-by-step list of instructions to solve the problem.

Road Example

Example 1

Imagine that you have the following image, which is a map of a road leading to the building shown in the picture.

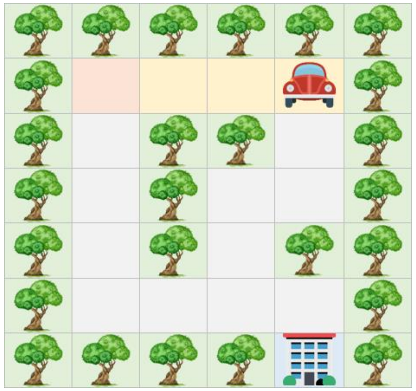
- There are a car and trees.
- The car cannot cross the trees.
- The road is divided into squares to calculate the steps of the car.
- Each square is considered as one step.



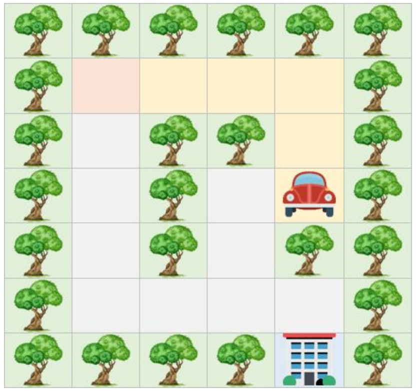
How can the car arrive at the building?

Road Example
Solution A

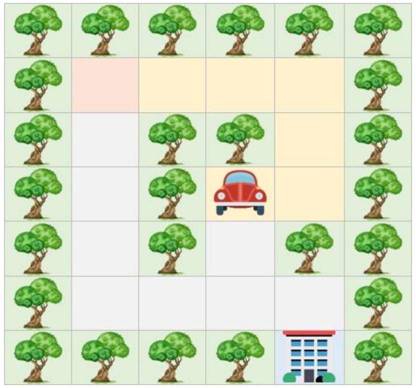
- Step 1: Move to the right three steps.
- Step 2: Move to down two steps.
- Step 3: Move to the left one step.
- Step 4: Move to down two steps.
- Step 5: Move to the right one step.
- Step 6: Move to down one step.



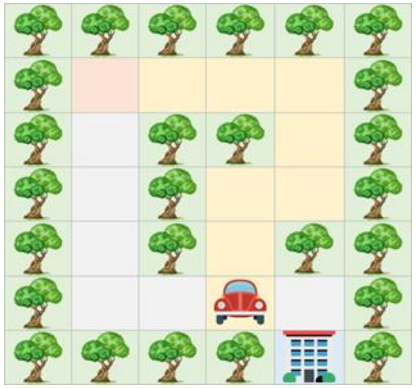
Step 1



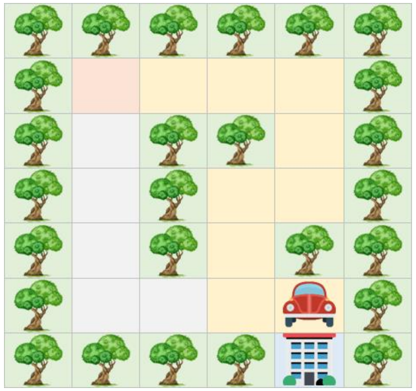
Step 2



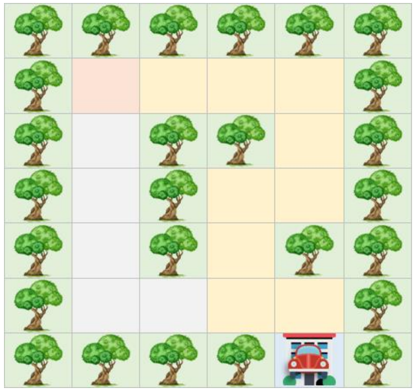
Step 3



Step 4



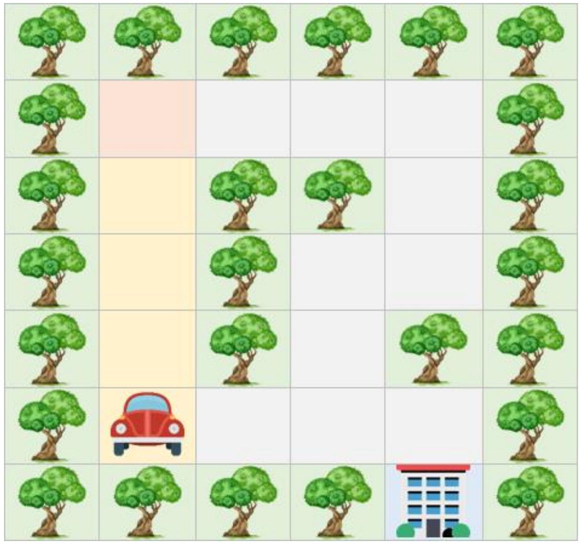
Step 5



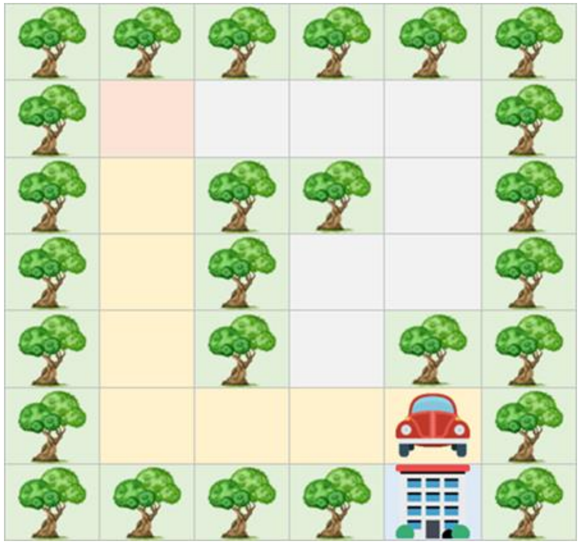
Step 6

Road Example
Solution B

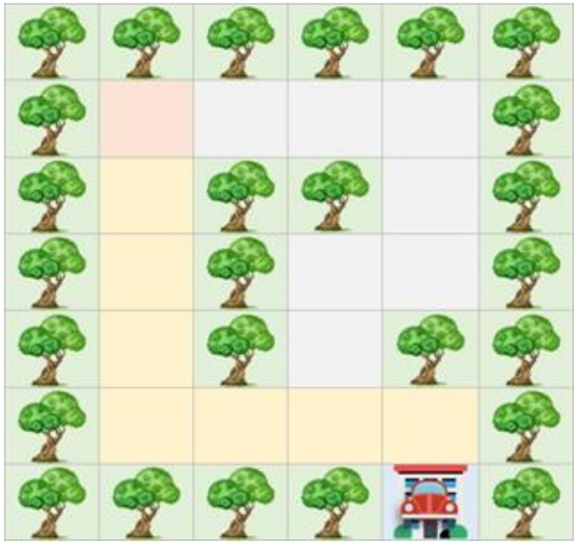
- Step 1: Move to down four steps.
- Step 2: Move to the right three steps.
- Step 3: Move to down one step.



Step 1



Step 2



Step 3

Road Example
Different Solutions

- As we can see that Solution A and Solution B are both correct solutions to the same problem, but there are differences in the complexity and efficiency of the solutions.

- The cost of Solution A is 10 steps while Solution B is 8 steps, so we consider Solution B as a better solution based on the number of steps.

- Reducing the number of steps in the previous example means reducing the amount of fuel needed by the vehicle and speeding up the arrival time.

Algorithms

- An algorithm is a set of obvious, logical, and sequential steps that solve a specific problem.
- To put it simply, the algorithm is like a recipe for preparing a specific food.
- Following the steps of the algorithm will end up solving the problem.

Area of a Rectangle Calculator

Example 2

Write an algorithm that can calculate the area of a rectangle. The width and the height of the rectangle should be taken from the user.

Note:

Area = Width × Height

Area of a Rectangle Calculator
Solution A

Solution A – Good:

1. Ask the user to enter Width
2. Ask the user to enter Height
3. Set Area to (Width × Height)
4. Display Area for the user

- As you can see in this solution, we have described the steps that are going to solve the problem.
- You can describe the steps in your own way, but your description of the steps should be obvious, logical, and sequential.

Area of a Rectangle Calculator
Solution B

Solution B - Bad:

1. Ask the user to enter Width
2. Ask the user to enter Height
3. Calculate Area
4. Display Area for the user

The reason for considering Solution B as a bad solution:

- Step 3 is not clear because it does not explain how we can calculate Area.
- So, this algorithm is bad because its steps are not obvious.

Area of a Rectangle Calculator
Solution C

Solution C - Bad:

1. Set Area to (Width × Height)
2. Ask the user to enter Width
3. Ask the user to enter Height
4. Display Area for the user

The reasons for considering Solution C as a bad solution:

- We don't know what Width and Height at the Step 1 are. In other words, Width and Height have not been defined before Step 1, so we cannot use them because they do not exist yet.
- What about Step 2 and Step 3? Width and Height are defined there!. After Step 2, Width does exist, but Height does not. After Step 3, Height does exist. Both Width and Height are available to be used at or after step 4.
- So, this algorithm is bad because its steps are not correctly sequential.

Area of a Rectangle Calculator
Solution D

Solution D - Bad:

1. Set Area to (Width × Height)
2. Display Area for the user

The reasons for considering Solution D as a bad solution:

- Step 1 tells us to multiply Width and Height, but we don't know what Width and Height are. Even, they have not been defined in any steps of the algorithm.
- So, this algorithm is bad because of the illogical step, which is using unknown things (Width and Height).

Area of a Rectangle Calculator
Solution E

Solution E - Bad:

1. Ask the user to enter Width
2. Ask the user to enter Height
3. Set Area to (Width × Height × 2)
4. Display Area for the user

The reasons for considering Solution E as a bad solution:

- This algorithm will give us a wrong value of the Area. For example, suppose that the user entered **4** for Width and **5** for Height. The correct value of the Area should be **20**, but this algorithm will display **40** as the value of the Area.

- The reason for giving the wrong value is how Step 3 calculates the Area. Step 3 calculates the Area by the incorrect equation (Width × Height × 2) instead of (Width × Height).

- So, this algorithm is bad because it has a logical problem, which is producing incorrect output (the value of the Area).

0.2. Program Design & Problem-Solving Techniques

- How Do We Write a Program?
- Problem-Solving Phase
- Implementation Phase

How Do We Write a Program?

- A Computer is not intelligent.
  - It cannot analyze a problem and come up with a solution.
  - A human (the programmer) must analyze the problem, develop the instructions for solving the problem, and then have the computer carry out the instructions.

- To write a program for a computer to follow, we must go through a two-phase process: problem solving and implementation.

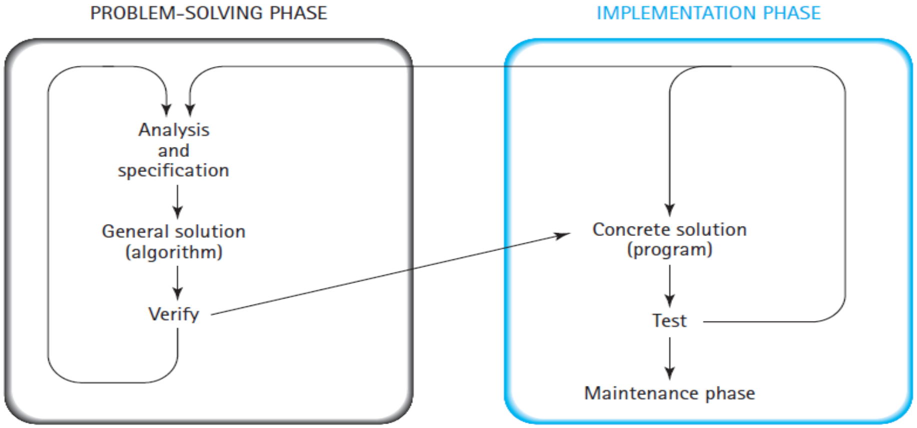

PROBLEM-SOLVING PHASE
IMPLEMENTATION PHASE
Analysis and specification
General solution (algorithm)
Verify
Concrete solution (program)
Test
Maintenance phase

Figure *Programming process*

Problem-Solving Phase

1. **Analysis and Specification** - Understand (define) the problem and what the solution must do.

2. **General Solution (Algorithm)** - Specify the required data types and the logical sequences of steps that solve the problem.

3. **Verify** - Follow the steps exactly to see if the solution really does solve the problem.

Implementation Phase

- **Concrete Solution (Program)** - Translate the algorithm (the general solution) into a programming language.
- **Test** - Have the computer follow the instructions.
  - Then manually check the results.
  - If you find errors, analyze the program and the algorithm to determine the source of the errors, and then make corrections.
- Once a program is tested, it enters into next phase (**Maintenance**).
- Maintenance requires modification of the program to meet changing requirements or to correct any errors that show up while using it.

0.3. Steps in Program Development

- Example 3: Simple Calculator

Steps in Program Development

1. Define the problem into three separate components:
   - Inputs
   - Processing steps to produce required outputs.
   - Outputs

Steps in Program Development

2. Outline the solution.
    - Decompose the problem to smaller steps.
    - Establish a solution outline.

3. Develop the outline into an algorithm.
    - The solution outline is now expanded into an algorithm.

Steps in Program Development

4. Test the algorithm for correctness.
   - Very important in the development of a program, but often forgotten.
   - Major logic errors can be detected and corrected at an early stage.

5. Code the algorithm into a specific programming language.

Steps in Program Development

6. Run the program on the computer.

   - This step uses a program compiler or interpreter, and programmer-designed test data to machine-test the code for
     - Syntax errors
     - Runtime errors
     - Logic errors

7. Document and maintain the program.

Simple Calculator
Example 3

Suppose that you are asked to write a calculator program that can sum and subtract two integer numbers. Write the program requirements, specifications and algorithm.

Simple Calculator
The Requirements

Suppose that you are asked to write a calculator program that can sum and subtract two integer numbers. Write the program requirements, specifications and algorithm.

The requirements:

- The user can enter an equation which consists of two numbers and a sign (- or +).
- The program should calculate the equation correctly and display the result for the user.
- The program can sum and subtract two integer numbers.

Simple Calculator
The Specifications

Suppose that you are asked to write a calculator program that can sum and subtract two integer numbers. Write the program requirements, specifications and algorithm.

The specifications:

- When the program runs, it will display a welcome message that says, 'Welcome to our Calculator".
- The program will then ask the user to enter the first number.
- The program will then ask the user to enter the second number.
- The program will then ask the user to select the sign (calculation operators) from this set (-,+).
- The program will then display the correct result of the calculation on the screen and end.

Simple Calculator
Designing a Solution

- After the steps of identifying the problem (the requirements and specifications), we should have a clear idea about what is going exactly to be achieved and solved.
- In this step, we are going to describe how the specifications can be achieved.
- This means that we need to design an algorithm that fulfills the specifications.
- We can design the algorithm via written steps or visualized steps using, for example, flowcharts.
- In the written steps, we can use simple sentences in English or some special notations and structures in something called "Pseudocode".

Simple Calculator
The Algorithm

Suppose that you are asked to write a calculator program that can sum and subtract two integer numbers. Write the program requirements, specifications and algorithm.

The algorithm:

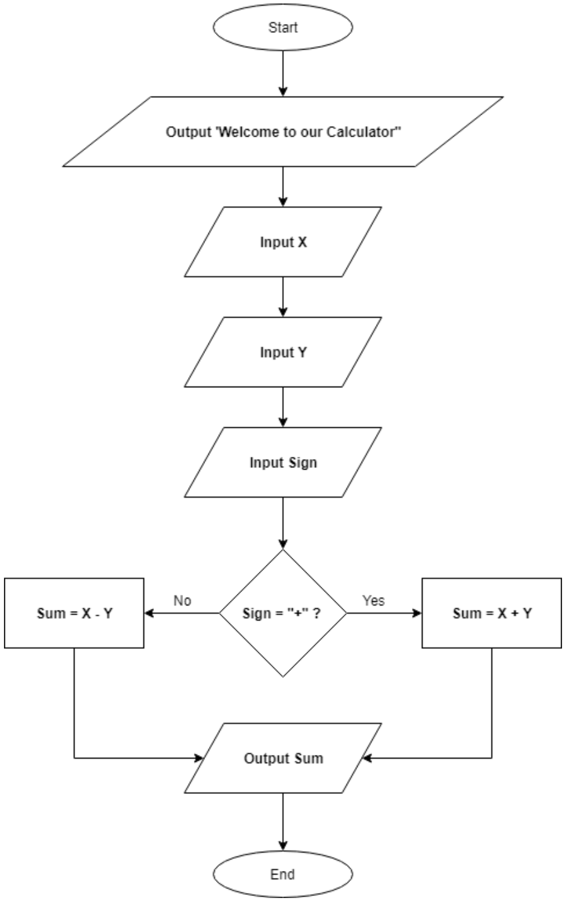
1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

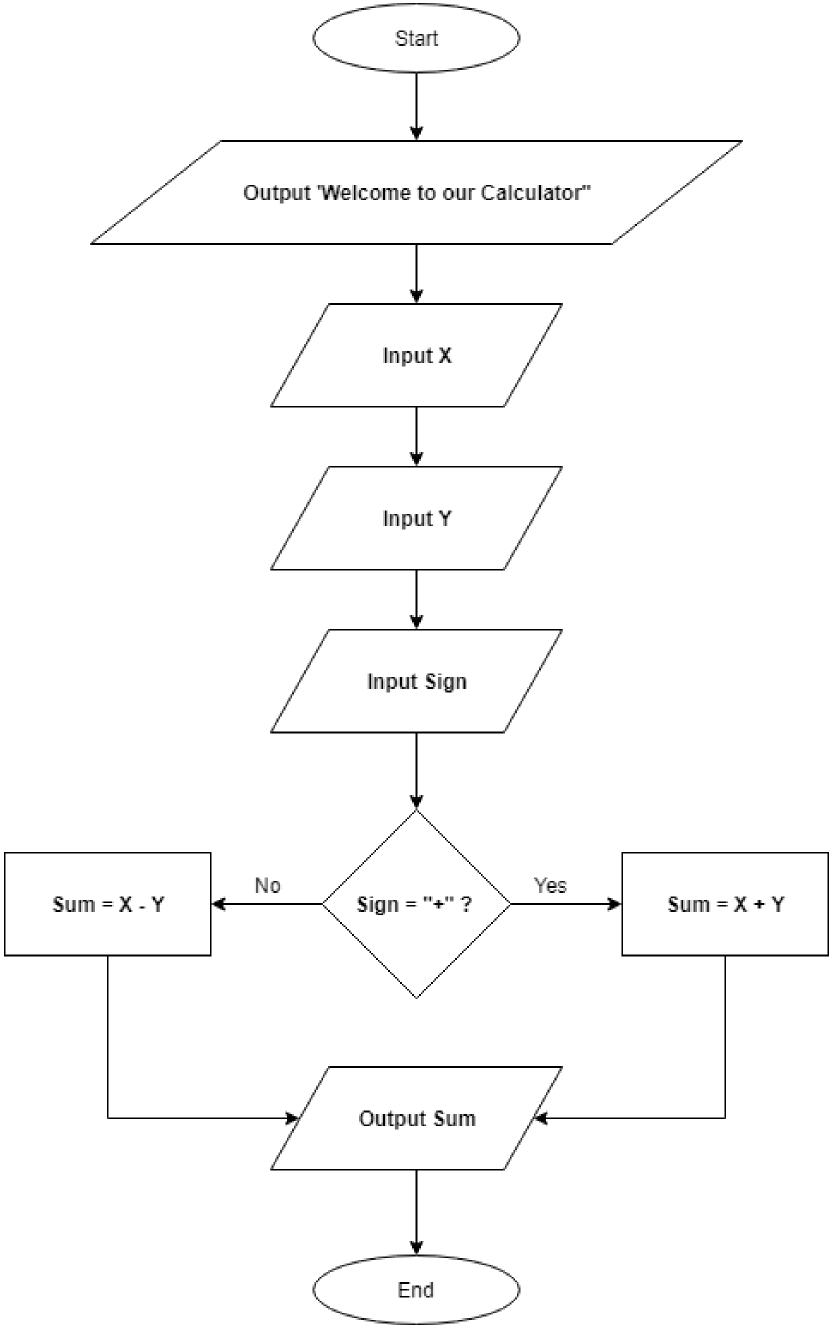
Simple Calculator
The Pseudocode and Flowchart

Suppose that you are asked to write a calculator program that can sum and subtract two integer numbers. Write the program requirements, specifications and algorithm.

The algorithm (pseudocode and flowchart):

```
print “Welcome to our Calculator”
X = input “Enter the first number:“
Y = input “Enter the second number:”
Sign = input "Select – or +"
if Sign is equal to "+" then:
    Sum = X + Y
else:
    Sum = X – Y
End if
print Sum
```





Start

Output 'Welcome to our Calculator"

Input X

Input Y

Input Sign

Sign = "+" ?

No

Sum = X - Y

Yes

Sum = X + Y

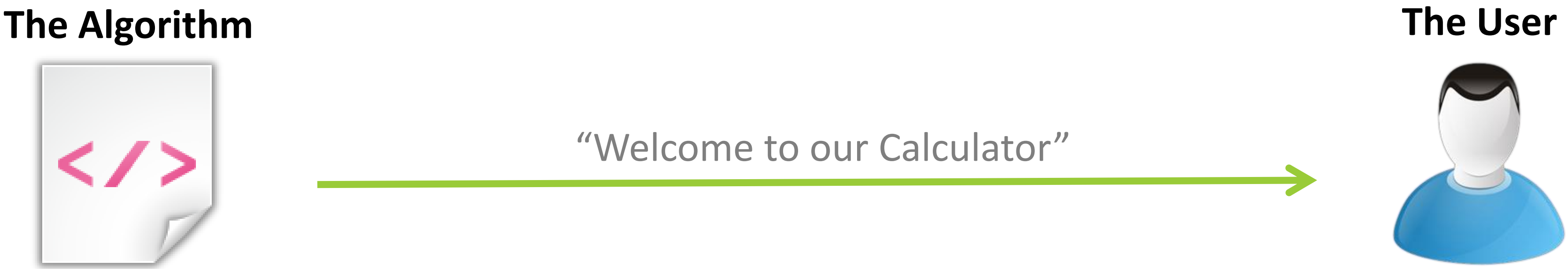
Output Sum

End

Simple Calculator
Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

Test 1



The Algorithm

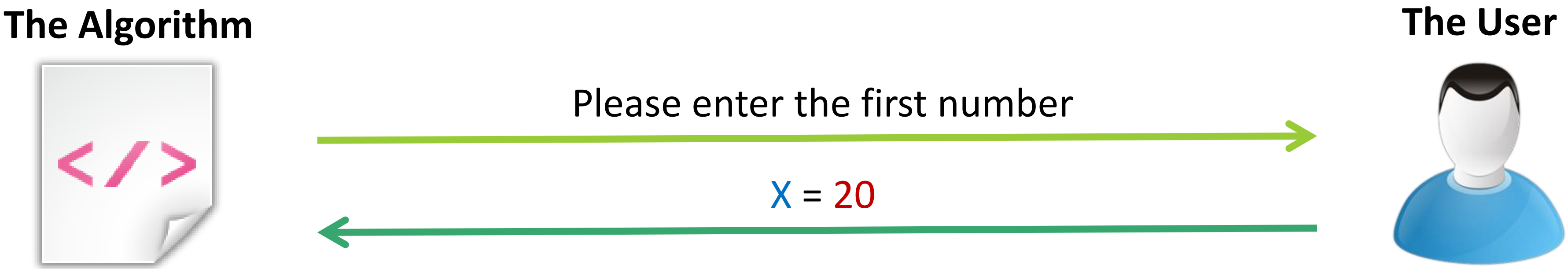
“Welcome to our Calculator”

The User

Simple Calculator
Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

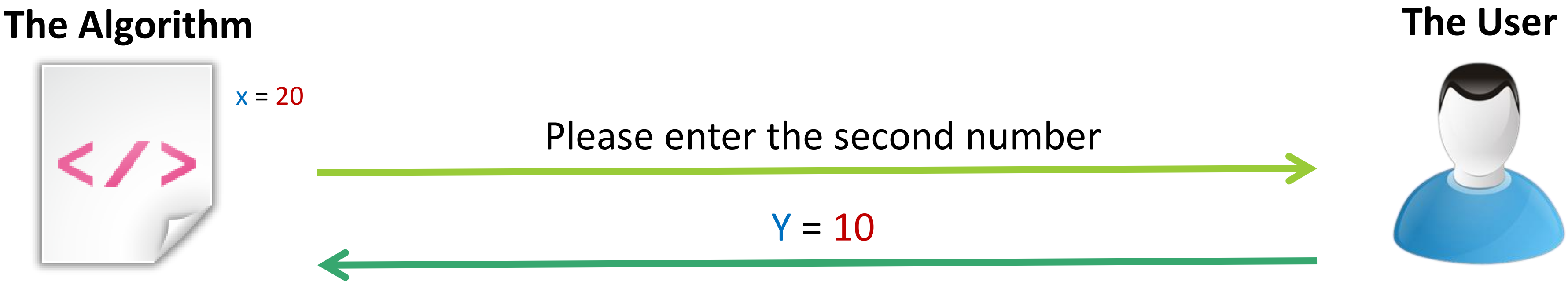
Test 1



Simple Calculator
Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

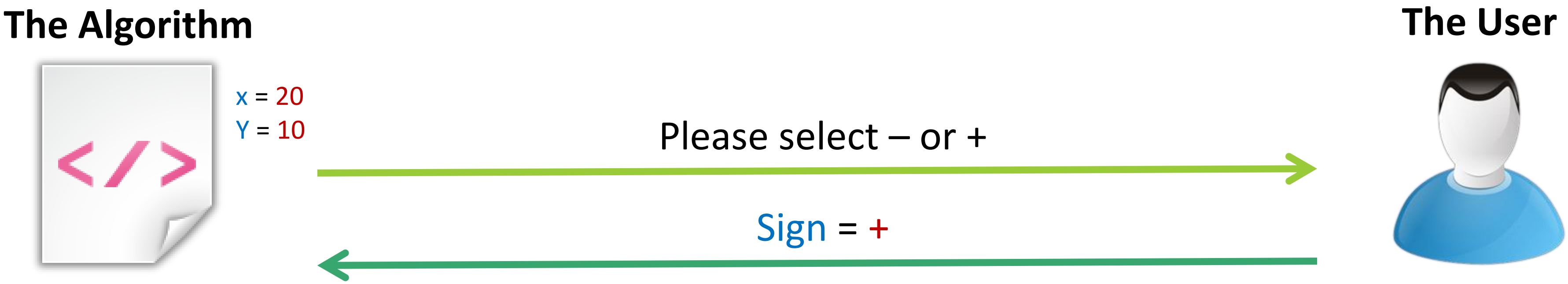
Test 1



Simple Calculator
Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

Test 1



The Algorithm

x = 20
Y = 10

Please select – or +

Sign = +

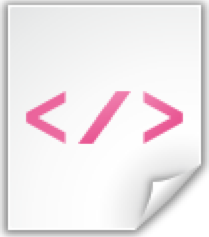
The User

Simple Calculator
Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

Test 1

The Algorithm



x = 20
Y = 10
Sign = +
Is Sign equal to “+”? **Yes**

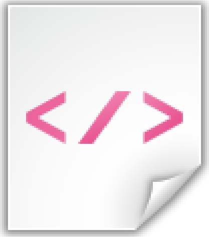
The User

Simple Calculator
Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

Test 1

The Algorithm



x = 20
Y = 10
Sign = +
Sum = X + Y = 20 + 10 = **30**

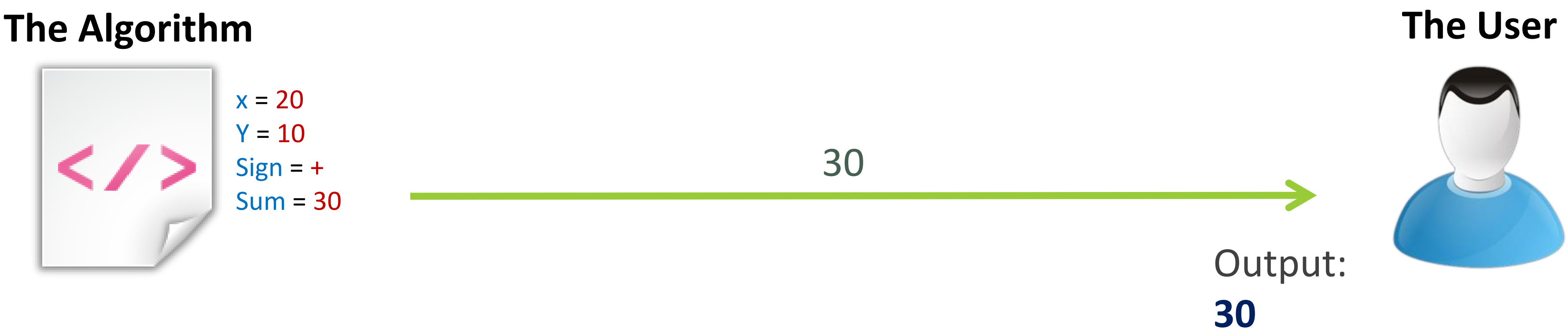
The User

Simple Calculator

Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

Test 1



The Algorithm

x = 20
Y = 10
Sign = +
Sum = 30

30

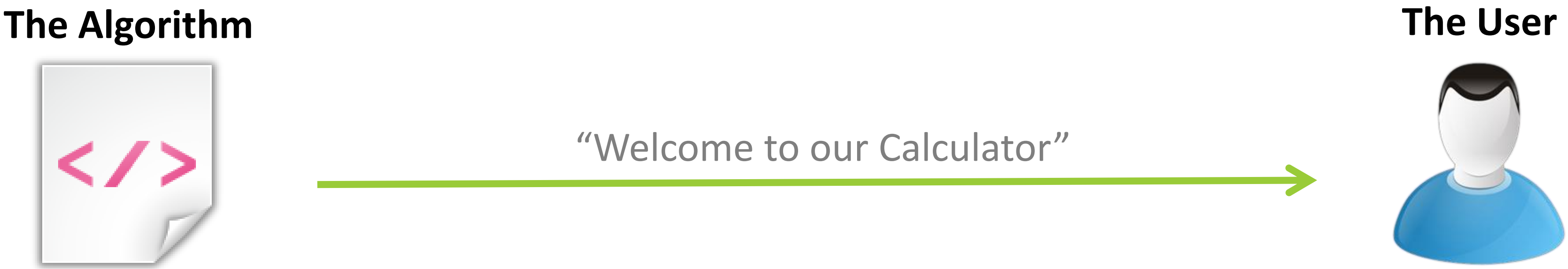
The User

Output:
30

Simple Calculator
Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

Test 2



The Algorithm

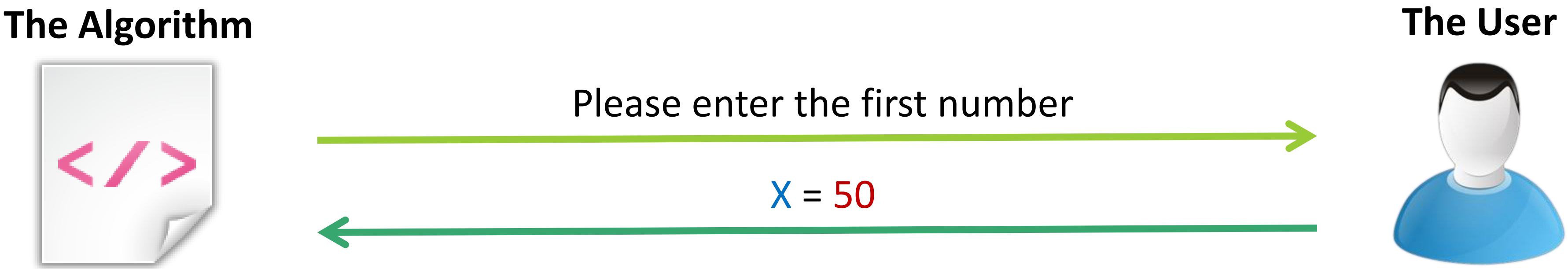
“Welcome to our Calculator”

The User

Simple Calculator
Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

Test 2



The Algorithm

The User

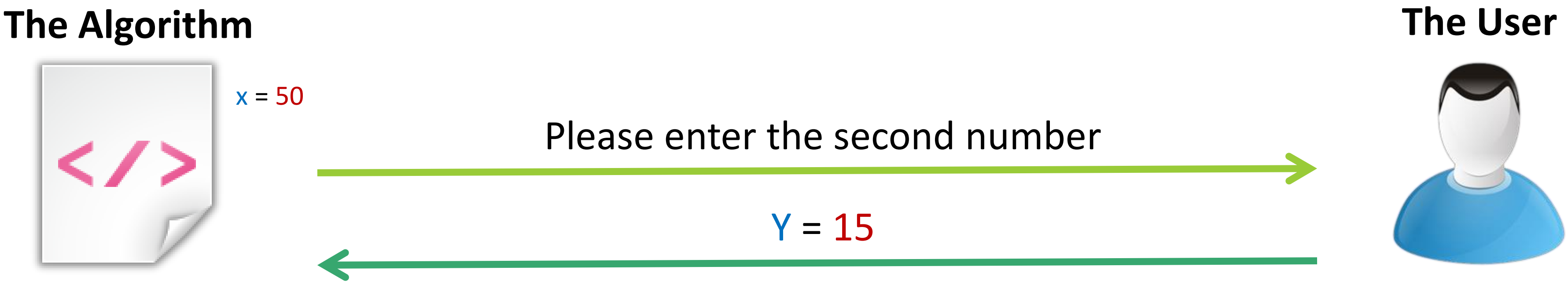
Please enter the first number

X = 50

Simple Calculator
Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

Test 2



The Algorithm

x = 50

Please enter the second number

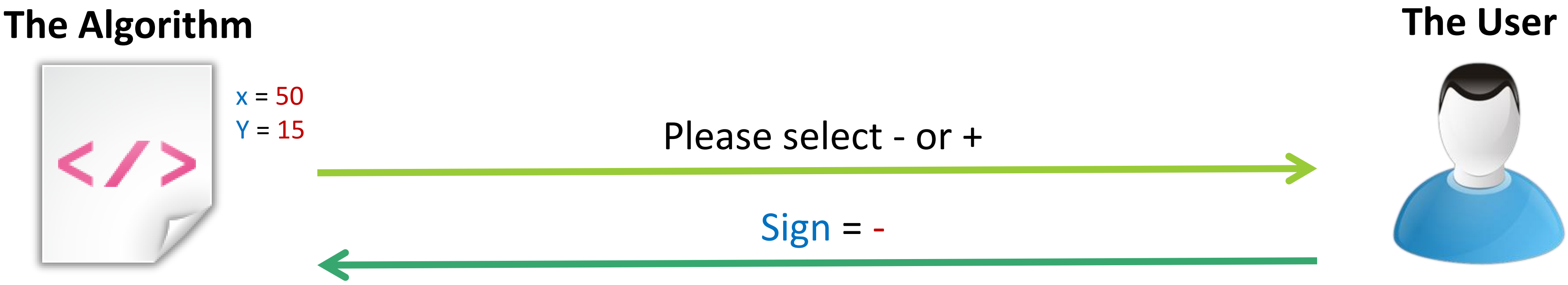
Y = 15

The User

Simple Calculator
Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

Test 2



The Algorithm

x = 50
Y = 15

Please select - or +

Sign = -

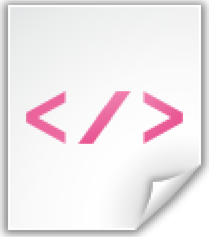
The User

Simple Calculator
Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

Test 2

The Algorithm



x = 50
Y = 15
Sign = -
Is Sign equal to “+”? **No**

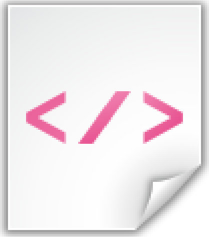
The User

Simple Calculator
Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

Test 2

The Algorithm



x = 50
Y = 15
Sign = -

Sum = X - Y = 50 - 15 = **35**

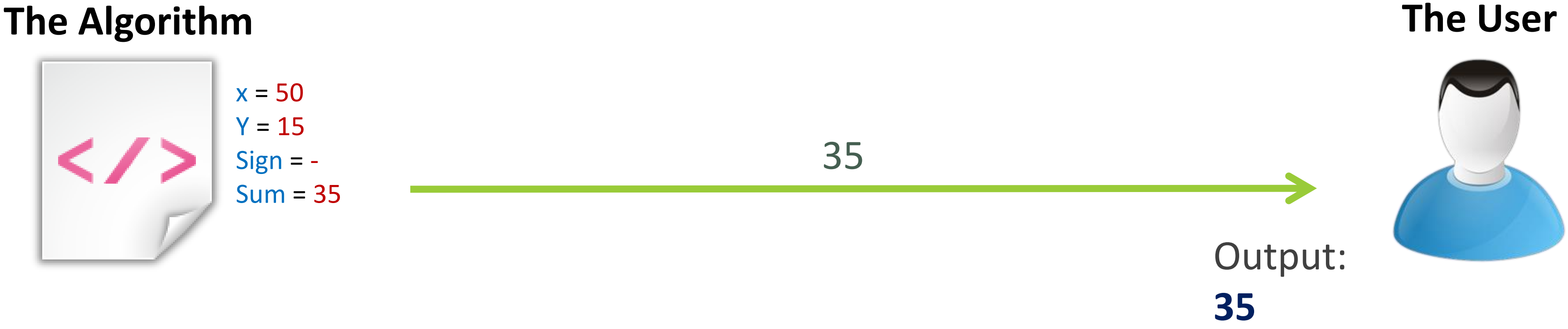
The User

Simple Calculator

Verifying The Algorithm

1. Display a welcome message that says, “Welcome to our Calculator".
2. Ask the user to enter the first number and save it to X.
3. Ask the user to enter the second number and save it to Y.
4. Ask the user to select the sign (-,+) and save it in Sign.
5. if Sign is equal to “+”, make Sum = X + Y
6. Otherwise, make Sum = X - Y
7. Display Sum

Test 2



The Algorithm

x = 50
Y = 15
Sign = -
Sum = 35

The User

Output:
35

0.4. Algorithms, Pseudocode, & Flowcharts

- Example 4: Determining a Student’s Final Grade
- Flowcharts
- Flowchart Symbols
- Example 5: Converting The Length
- Example 6: Area of a Rectangle Calculator

Algorithm, Pseudocode, & Flowcharts

- What is an algorithm?
  - A step-by-step series of instructions in order to perform a specific task.
  - An algorithm must:
    - Be lucid (clear), precise and unambiguous.
    - Give the correct solution in all cases, and eventually end.
- What is pseudocode?
  - It is English that looks similar to code
    - But it is not actual code (only looks a little similar) .
    - Think of pseudocode as a way of expressing your algorithm.
- What is a flowchart?
  - A graphical representation of the sequence of operations in an information system or program.

Algorithm, Pseudocode, & Flowcharts

- For Clarity:
  - An algorithm is a series of steps you take to solve a problem, just like a recipe is a series of steps you take to make a food!
  - Now, we express our algorithms in many ways:
    - **Pseudocode**: this is not "real code", but a slightly more formal way of writing the algorithmic steps
      - As an example, maybe the programmer does not know the language he/she will use. Therefore, they just write pseudocode during Problem-Solving Phase.
    - **Flowchart**: this is a graphical representation of the algorithm
    - **Actual code**: this is during the Implementation Phase
      - Python, Java, C++, C, etc

Determining a Student's Final Grade

Example 4

Write an algorithm and pseudocode to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

Determining a Student’s Final Grade
Algorithm

Write an algorithm and pseudocode to determine a student’s final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

Algorithm:

```
1. Ask the user to enter 4 marks (Mark1, Mark2, Mark3, Mark4)
2. Calculate the marks average (Avg) by summing marks and it dividing by 4
3. If average (Avg) is greater than or equal 60
4.     Print “Pass”
5. Else
6.     Print “Fail”
7. End if
```

Determining a Student’s Final Grade
Pseudocode

Write an algorithm and pseudocode to determine a student’s final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

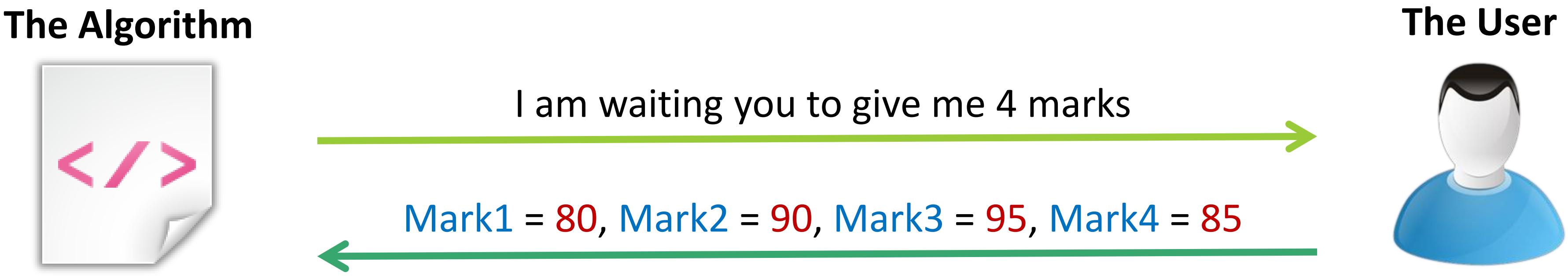
Pseudocode:

```
input Mark1, Mark2, Mark3, Mark4
Avg = (Mark1 + Mark2 + Mark3 + Mark4) / 4
if Avg >= 60:
        print “Pass”
else:
        print “Fail”
End if
```

Determining a Student’s Final Grade
Verifying The Algorithm

```
input Mark1, Mark2, Mark3, Mark4
Avg = (Mark1 + Mark2 + Mark3 + Mark4) / 4
if Avg >= 60:
        print “Pass”
else:
        print “Fail”
End if
```

Test 1



The Algorithm

The User

I am waiting you to give me 4 marks

Mark1 = 80, Mark2 = 90, Mark3 = 95, Mark4 = 85

Determining a Student’s Final Grade
Verifying The Algorithm

```
input Mark1, Mark2, Mark3, Mark4
Avg = (Mark1 + Mark2 + Mark3 + Mark4) / 4
if Avg >= 60:
    print “Pass”
else:
    print “Fail”
End if
```

Test 1

The Algorithm

Mark1 = 80, Mark2 = 90, Mark3 = 95, Mark4 = 85

Avg = (80 + 90 + 95 + 85) / 4 = 350 / 4 = 87.5

The User

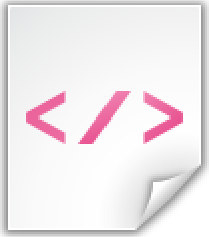
Determining a Student’s Final Grade

Verifying The Algorithm

```
input Mark1, Mark2, Mark3, Mark4
Avg = (Mark1 + Mark2 + Mark3 + Mark4) / 4
if Avg >= 60:
        print “Pass”
else:
        print “Fail”
End if
```

Test 1

The Algorithm



Mark1 = 80, Mark2 = 90, Mark3 = 95, Mark4 = 85

Avg = 87.5

Avg >= 60 = 87.5 >= 60 = Yes

The User

Determining a Student’s Final Grade

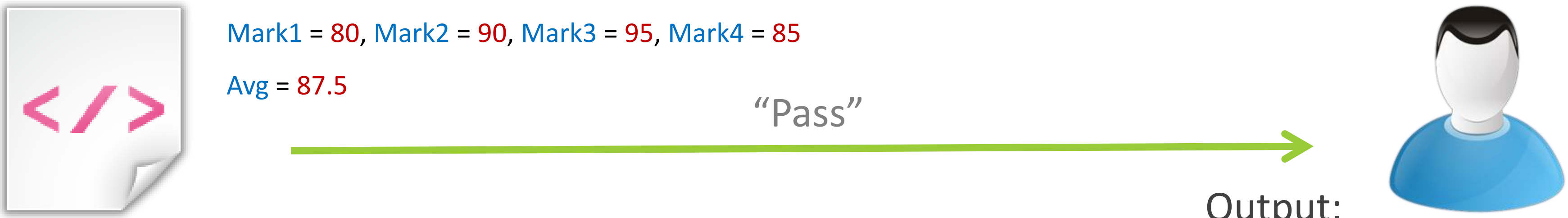
Verifying The Algorithm

```
input Mark1, Mark2, Mark3, Mark4
Avg = (Mark1 + Mark2 + Mark3 + Mark4) / 4
if Avg >= 60:
        print “Pass”
else:
        print “Fail”
End if
```

Test 1

The Algorithm

The User



Mark1 = 80, Mark2 = 90, Mark3 = 95, Mark4 = 85

Avg = 87.5

Output:

Pass

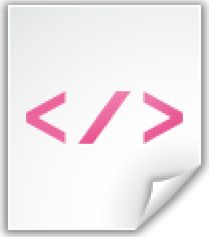
Determining a Student’s Final Grade

Verifying The Algorithm

```
input Mark1, Mark2, Mark3, Mark4
Avg = (Mark1 + Mark2 + Mark3 + Mark4) / 4
if Avg >= 60:
        print “Pass”
else:
        print “Fail”
End if
```

Test 1

The Algorithm



Mark1 = 80, Mark2 = 90, Mark3 = 95, Mark4 = 85

Avg = 87.5

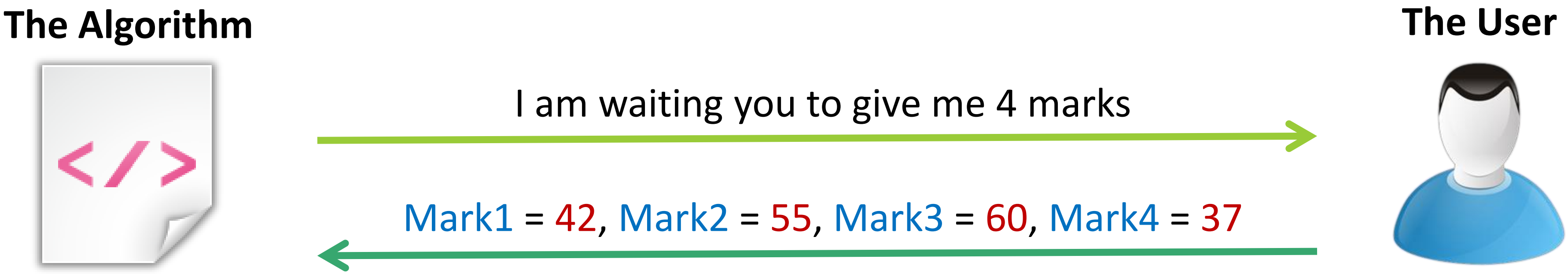
The User

Output:
Pass

Determining a Student’s Final Grade
Verifying The Algorithm

```
input Mark1, Mark2, Mark3, Mark4
Avg = (Mark1 + Mark2 + Mark3 + Mark4) / 4
if Avg >= 60:
    print “Pass”
else:
    print “Fail”
End if
```

Test 2



The Algorithm

The User

I am waiting you to give me 4 marks

Mark1 = 42, Mark2 = 55, Mark3 = 60, Mark4 = 37

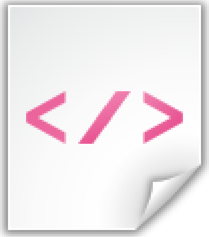
Determining a Student’s Final Grade

Verifying The Algorithm

```
input Mark1, Mark2, Mark3, Mark4
Avg = (Mark1 + Mark2 + Mark3 + Mark4) / 4
if Avg >= 60:
        print “Pass”
else:
        print “Fail”
End if
```

Test 2

The Algorithm



Mark1 = 42, Mark2 = 55, Mark3 = 60, Mark4 = 37

Avg = (42 + 55 + 60 + 37) / 4 = 194 / 4 = 48.5

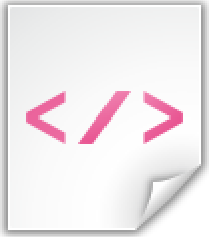
The User

Determining a Student’s Final Grade
Verifying The Algorithm

```
input Mark1, Mark2, Mark3, Mark4
Avg = (Mark1 + Mark2 + Mark3 + Mark4) / 4
if Avg >= 60:
        print “Pass”
else:
        print “Fail”
End if
```

Test 2

The Algorithm



Mark1 = 42, Mark2 = 55, Mark3 = 60, Mark4 = 37

Avg = 48.5

Avg >= 60 = 48.5 >= 60 = No

The User

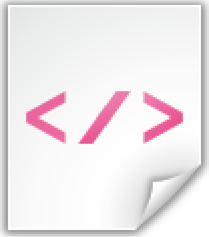
Determining a Student’s Final Grade

Verifying The Algorithm

```
input Mark1, Mark2, Mark3, Mark4
Avg = (Mark1 + Mark2 + Mark3 + Mark4) / 4
if Avg >= 60:
        print “Pass”
else:
        print “Fail”
End if
```

Test 2

The Algorithm



Mark1 = 42, Mark2 = 55, Mark3 = 60, Mark4 = 37

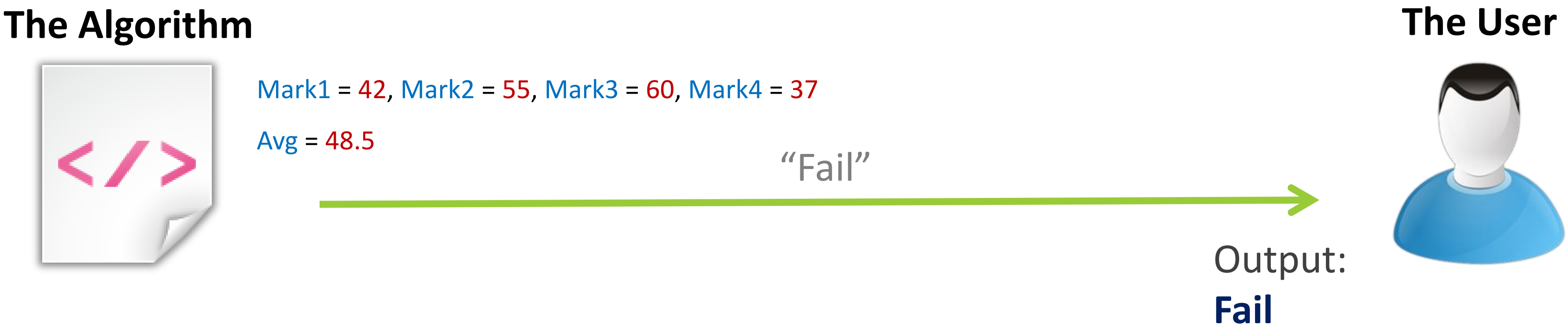
Avg = 48.5

The User

Determining a Student’s Final Grade
Verifying The Algorithm

```
input Mark1, Mark2, Mark3, Mark4
Avg = (Mark1 + Mark2 + Mark3 + Mark4) / 4
if Avg >= 60:
        print “Pass”
else:
        print “Fail”
End if
```

Test 2



The Algorithm

Mark1 = 42, Mark2 = 55, Mark3 = 60, Mark4 = 37

Avg = 48.5

The User

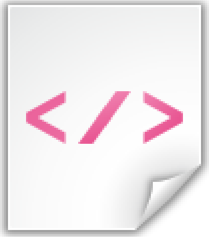
Output:
Fail

Determining a Student’s Final Grade
Verifying The Algorithm

```
input Mark1, Mark2, Mark3, Mark4
Avg = (Mark1 + Mark2 + Mark3 + Mark4) / 4
if Avg >= 60:
    print “Pass”
else:
    print “Fail”
End if
```

Test 2

The Algorithm



Mark1 = 42, Mark2 = 55, Mark3 = 60, Mark4 = 37

Avg = 48.5

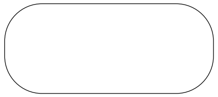
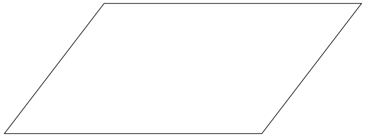
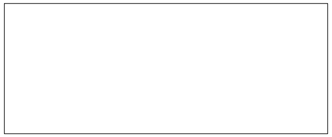
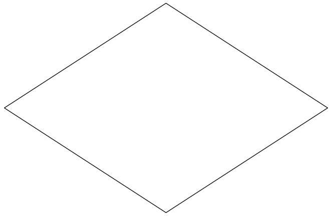
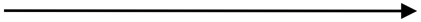
The User

Output:
Fail

Flowchart

- A graphical representation of the sequence of operations in an information system or program.
- Program flowcharts show the sequence of instructions in a single program or subroutine.
  - show logic of an algorithm
  - emphasize individual steps and their interconnections
  - e.g. control flow from one action to the next
- Different symbols are used to draw each type of flowchart.

Flowchart Symbols

Name	**Symbol**	**Use in Flowchart**
Oval		Denotes the beginning or end of the program.
Parallelogram		Denotes an input / output operations.
Rectangle		Denotes a process to be carried out. For example: addition, subtraction, and division.
Diamond		Denotes a decision or branch to be made The program should continue along one of two routes (Ex. If/Then/Else)
Flow line		Denotes the direction of logic flow in the program

Flowcharts

- Are flowcharts really necessary or helpful?
  - In the real world, programs are not only 1000 lines.
  - Programs are hundreds of thousands of lines of code.
    - Even Millions of lines of code.
  - Could you use only English to describe your program?
    - Sure you could, but you would end up with a book!
  - Therefore, think of flowcharts as an easier, clearer way to quickly understand what a program is doing.

Flowcharts

- Are flowcharts really necessary or helpful?
  - So in summary, yes, they are helpful.

- That said, most of the programs we show you over the next few weeks are smaller programs.
  - Do you really need a flowchart for a small program?
  - Probably not.
  - However, students should get into the habit of making flowcharts with smaller, easier programs.
  - Then, it will be easy to do for larger programs.

Converting The Length
Example 5

Write an Algorithm, Pseudocode, and draw a flowchart to convert the length in feet to centimeter.

Algorithm:

1. Input the length in feet (LFT)
2. Calculate the length in cm (LCM) by multiplying LFT with 30
3. Print length in cm (LCM)

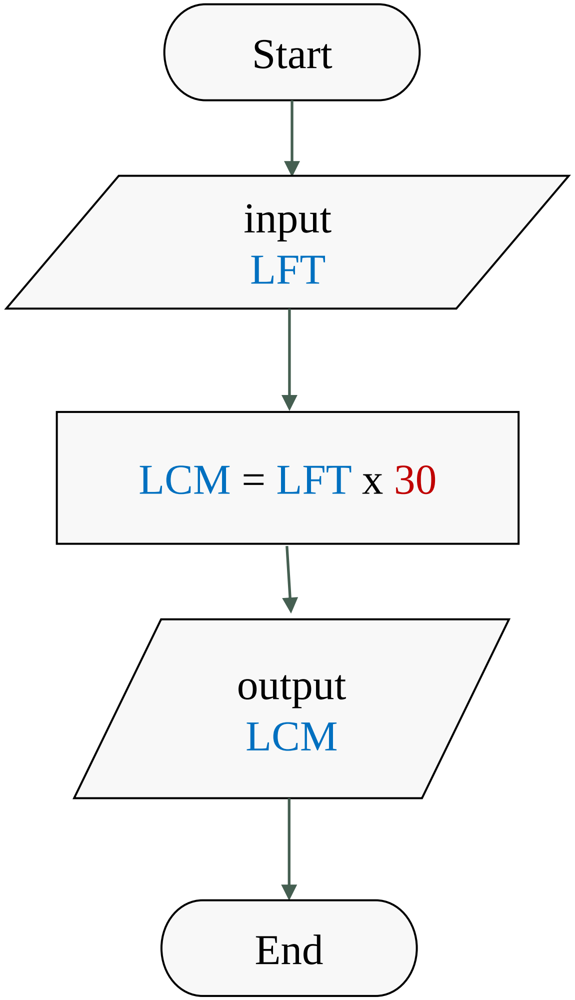
Converting The Length

The Algorithm

Pseudocode:

```
1. LFT  = input "Length in feet"
2. LCM = LFT x 30
3. print LCM
```

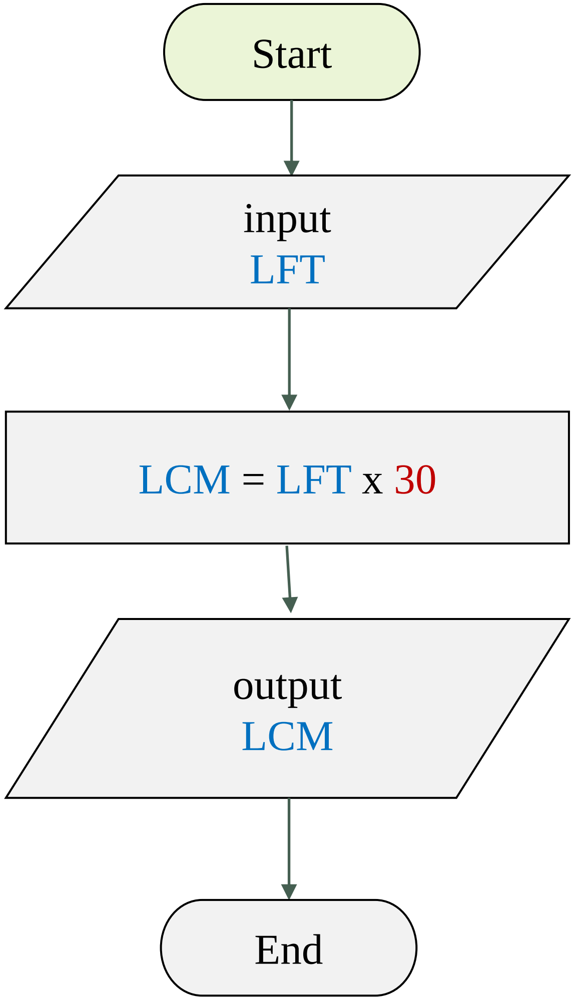
Flowchart:



Converting The Length

Verifying The Algorithm

The Algorithm



Start

input
LFT

LCM = LFT x 30

output
LCM

End

Test 1

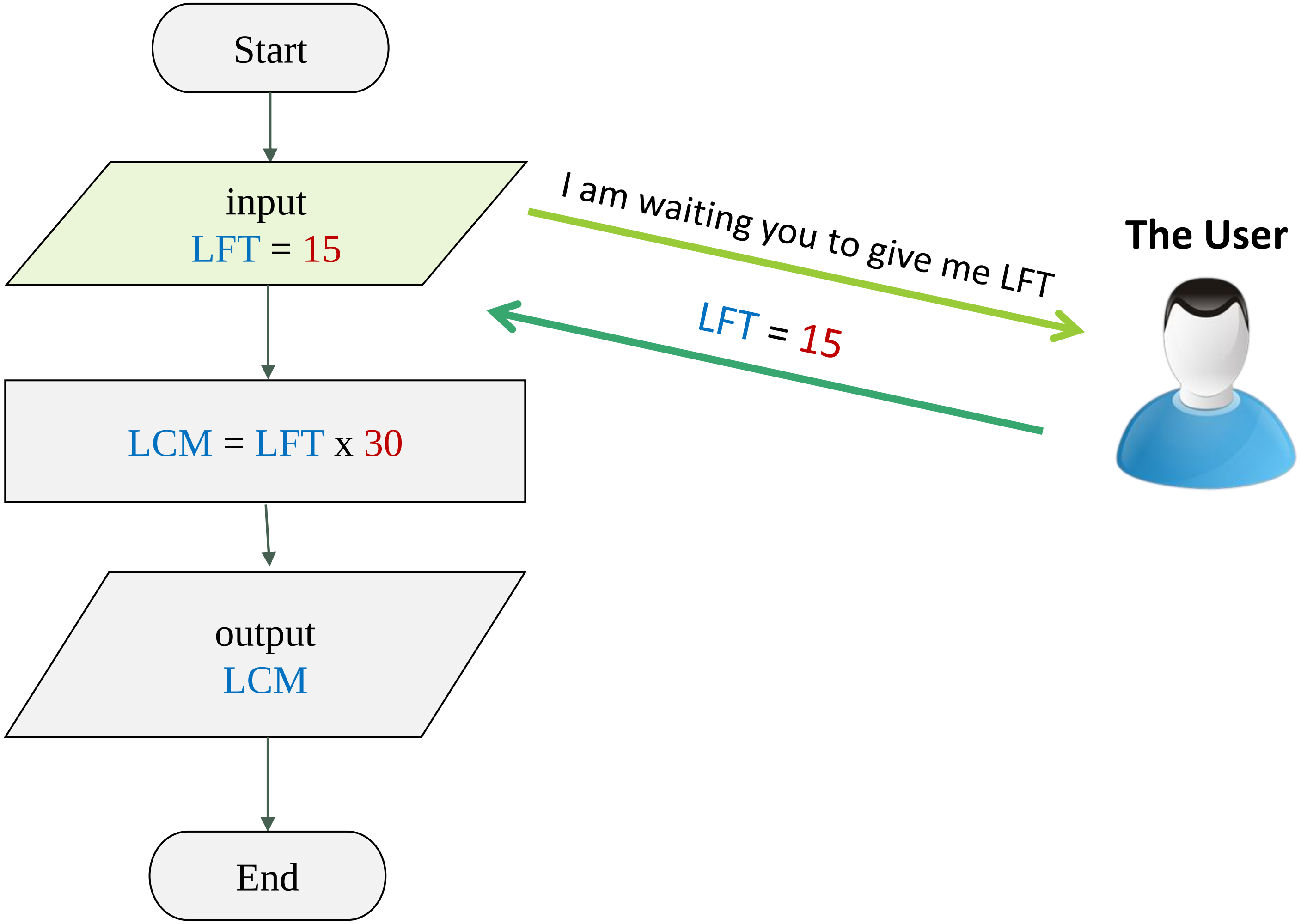
The User

Converting The Length

Verifying The Algorithm

Test 1

The Algorithm



Start

input
LFT = 15

LCM = LFT x 30

output
LCM

End

I am waiting you to give me LFT

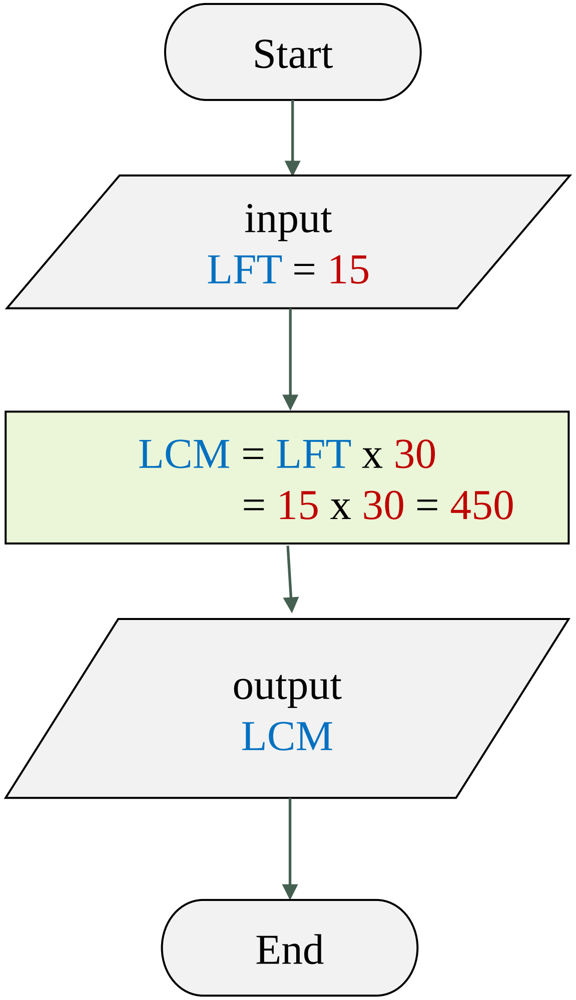
LFT = 15

The User

Converting The Length

Verifying The Algorithm

The Algorithm



Start

input
LFT = 15

LCM = LFT x 30
= 15 x 30 = 450

output
LCM

End

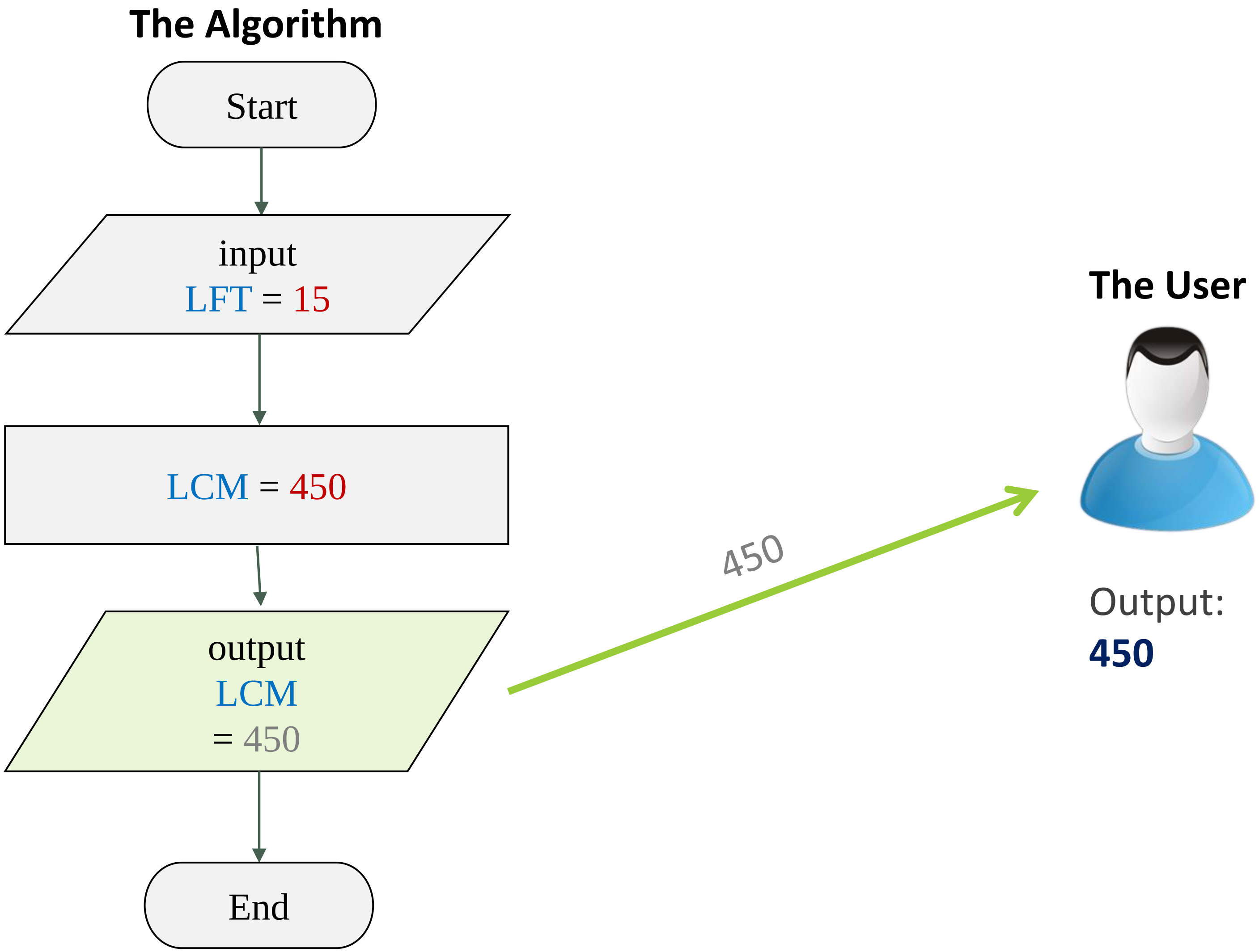
Test 1

The User

Converting The Length

Verifying The Algorithm

Test 1



The Algorithm

Start

input
LFT = 15

LCM = 450

output
LCM
= 450

End

450

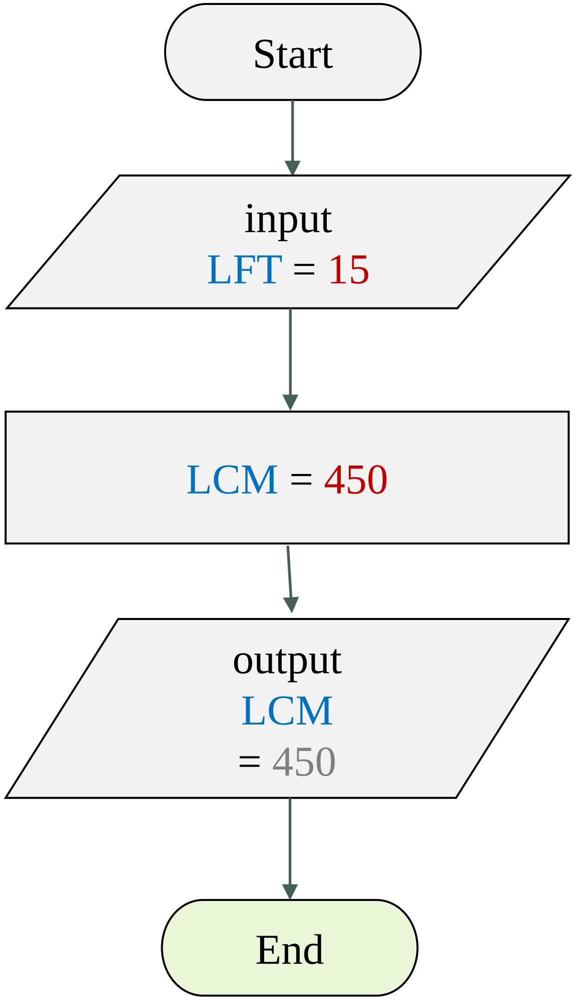
The User

Output:
450

Converting The Length

Verifying The Algorithm

The Algorithm



Start

input
LFT = 15

LCM = 450

output
LCM
= 450

End

Test 1

The User

Output:
450

Area of a Rectangle Calculator
Example 6

Write an Algorithm, Pseudocode, and draw a flowchart that will read the two sides of a rectangle and calculate its area.

Algorithm:

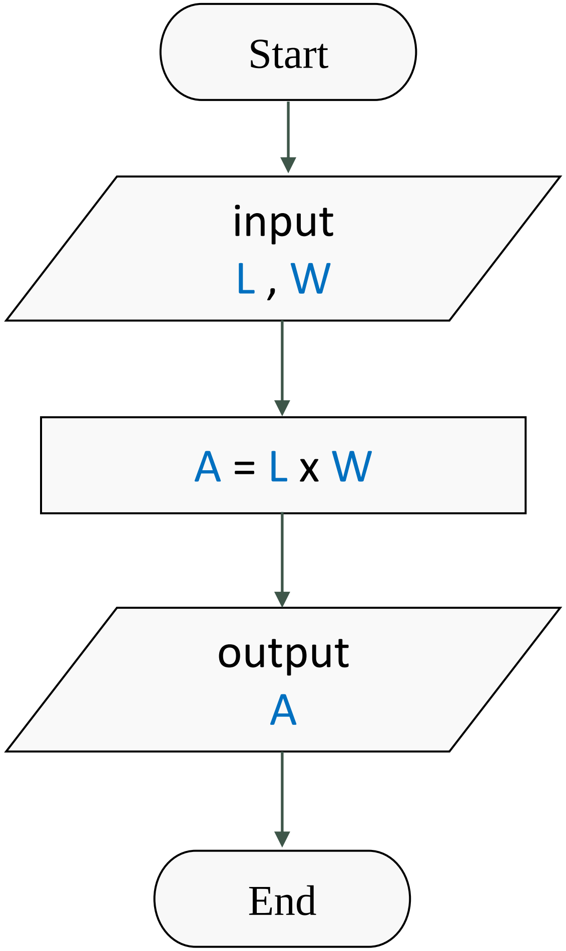
1. Input the Length (L) and width (W) of a rectangle
2. Calculate the area (A) by multiplying L with W
3. Print A

Area of a Rectangle Calculator
The Algorithm

Pseudocode:

```
input L, W
A = L x W
print A
```

Flowchart:



Start

input L , W

A = L x W

output A

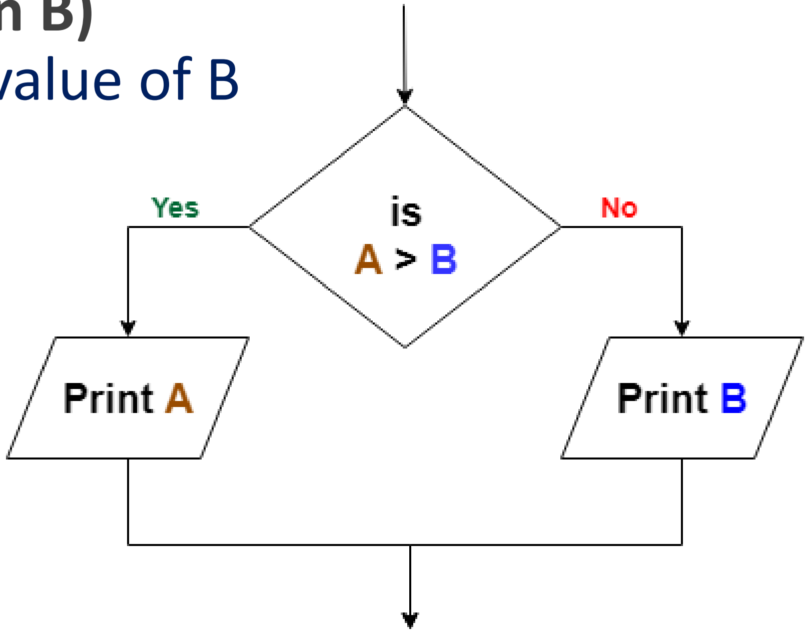
End

0.5. Decision Structures

- If–then–else Structure
- Relational Operators
- Example 7: Determining The Largest Value

Decision Structures

- The expression A > B is a logical expression
- It describes a **condition** we want to test
- **if A > B is true (if A is greater than B)**
  we take the action on left: print the value of A
- **if A > B is false (if A is not greater than B)**
  we take the action on right: print the value of B
- Note: Print = Output



If–then–else Structure

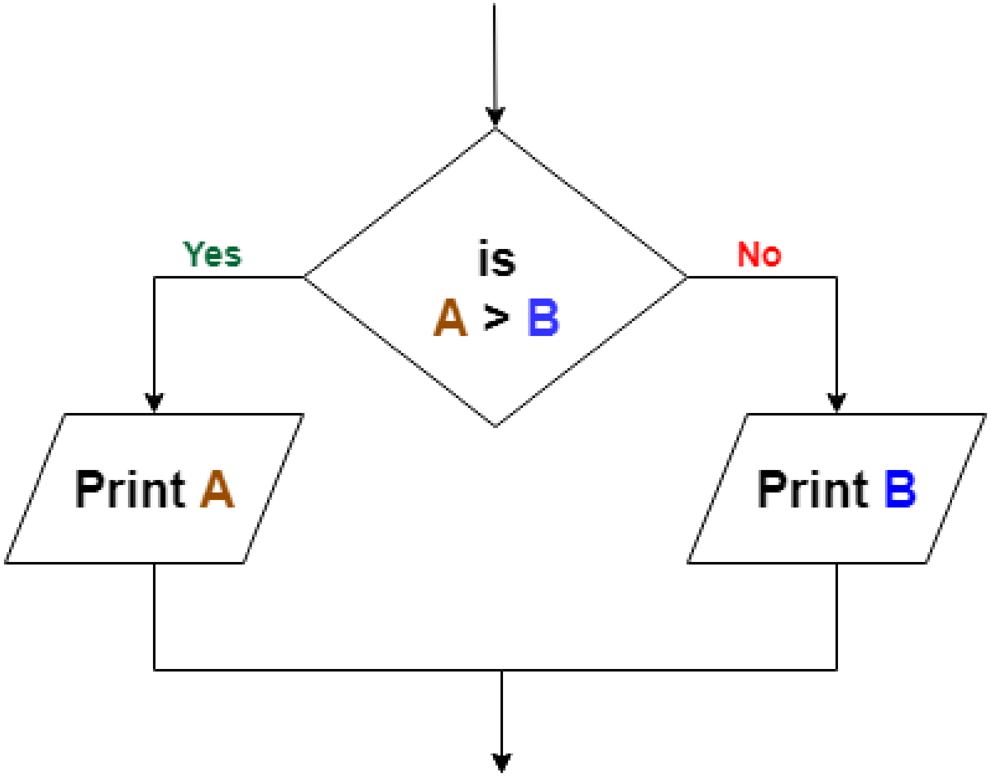
- The structure is as follows

```
if condition then
        true alternative
else
        false alternative
End if
```

If–then–else Structure

- The algorithm for the flowchart is as follows:

```
if A > B then
        print A
else
        print B
End if
```



Relational Operators

Relational Operators	
Operator	**Description**
>	Greater than
<	Less than
==	Equal to
≥ Or >=	Greater than or equal to
≤ Or <=	Less than or equal to
≠ Or !=	Not equal to

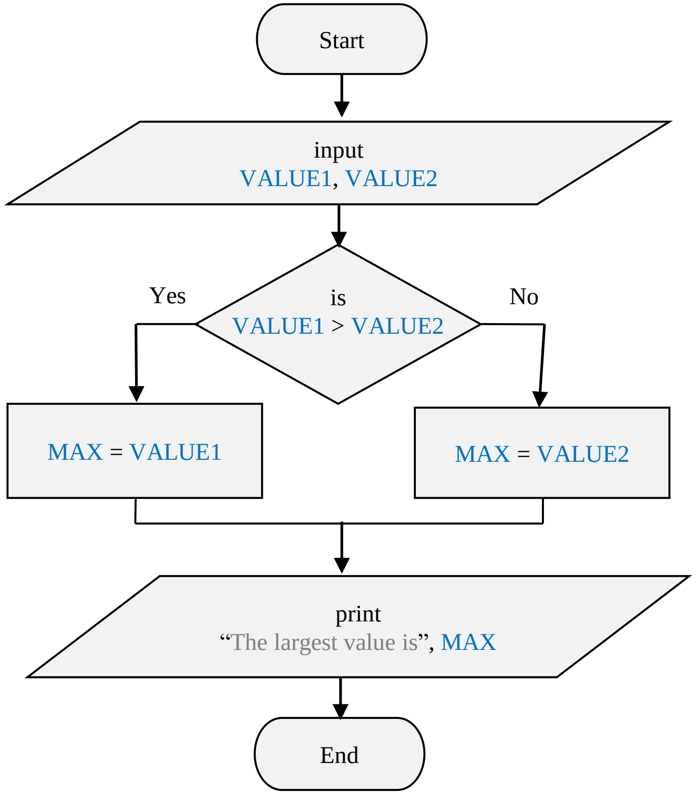
Determining The Largest Value

Example 7

Write a Pseudocode that reads two values, determines the largest value and prints the largest value with an identifying message.

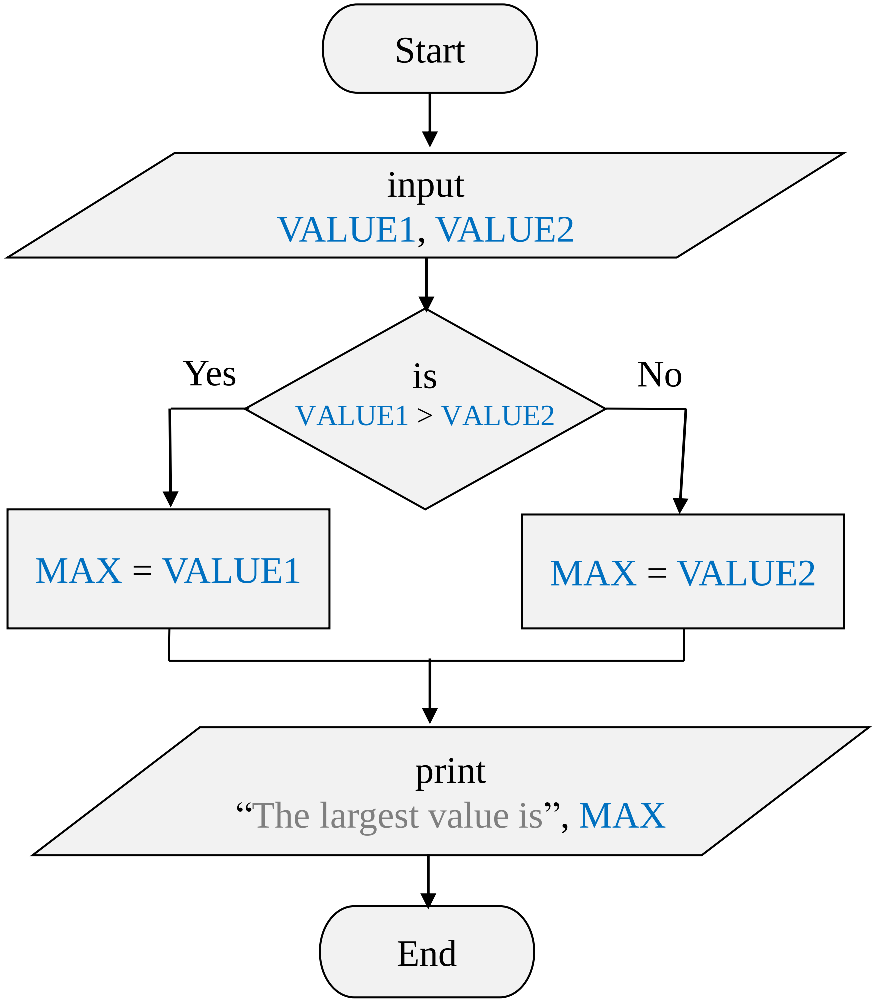
Pseudocode:

```
Input VALUE1, VALUE2
if (VALUE1 > VALUE2) then
        MAX = VALUE1
else
        MAX = VALUE2
endif
print "The largest value is", MAX
```



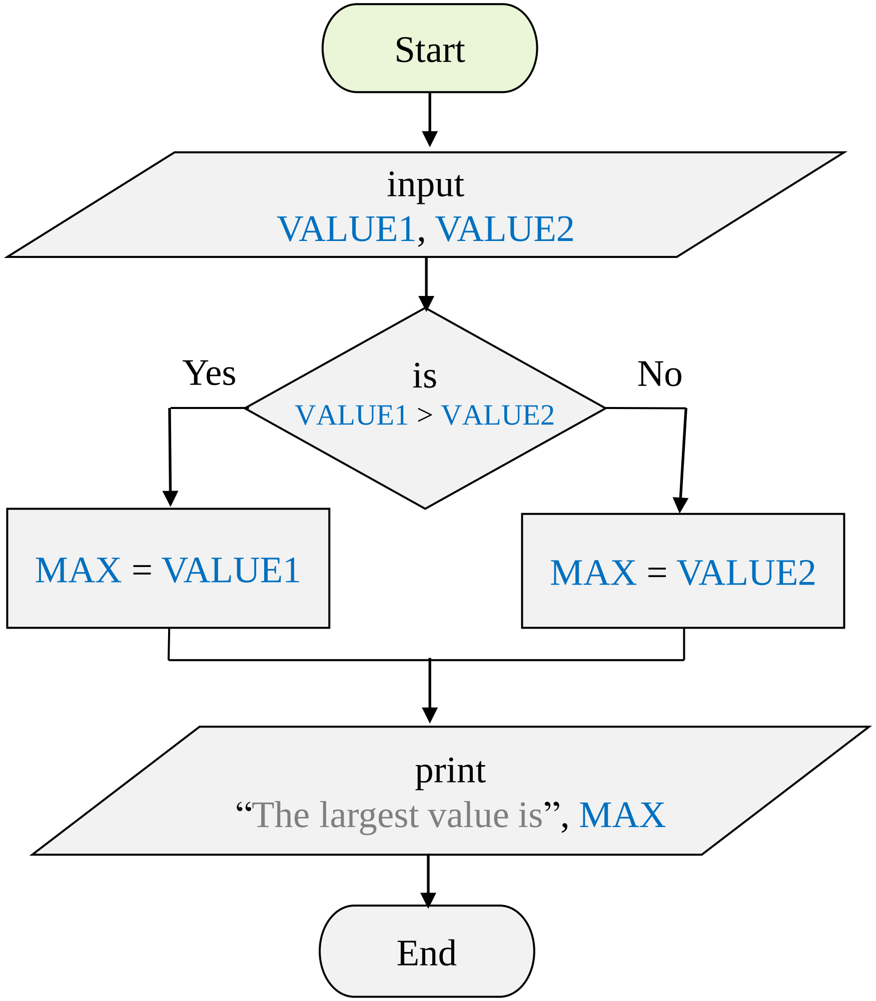
Determining The Largest Value

The Algorithm



Determining The Largest Value
Verifying The Algorithm

The Algorithm

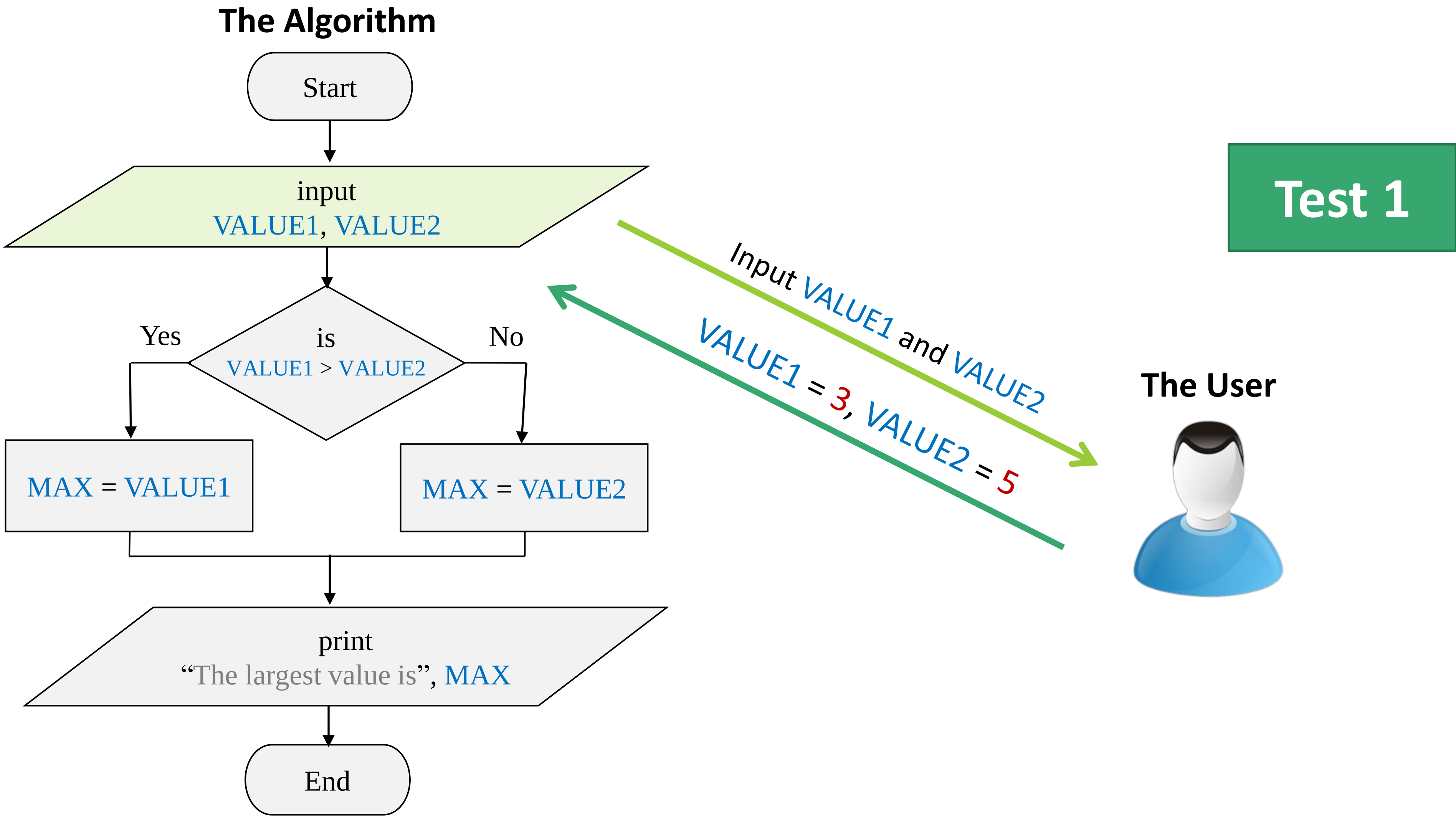


Test 1

The User

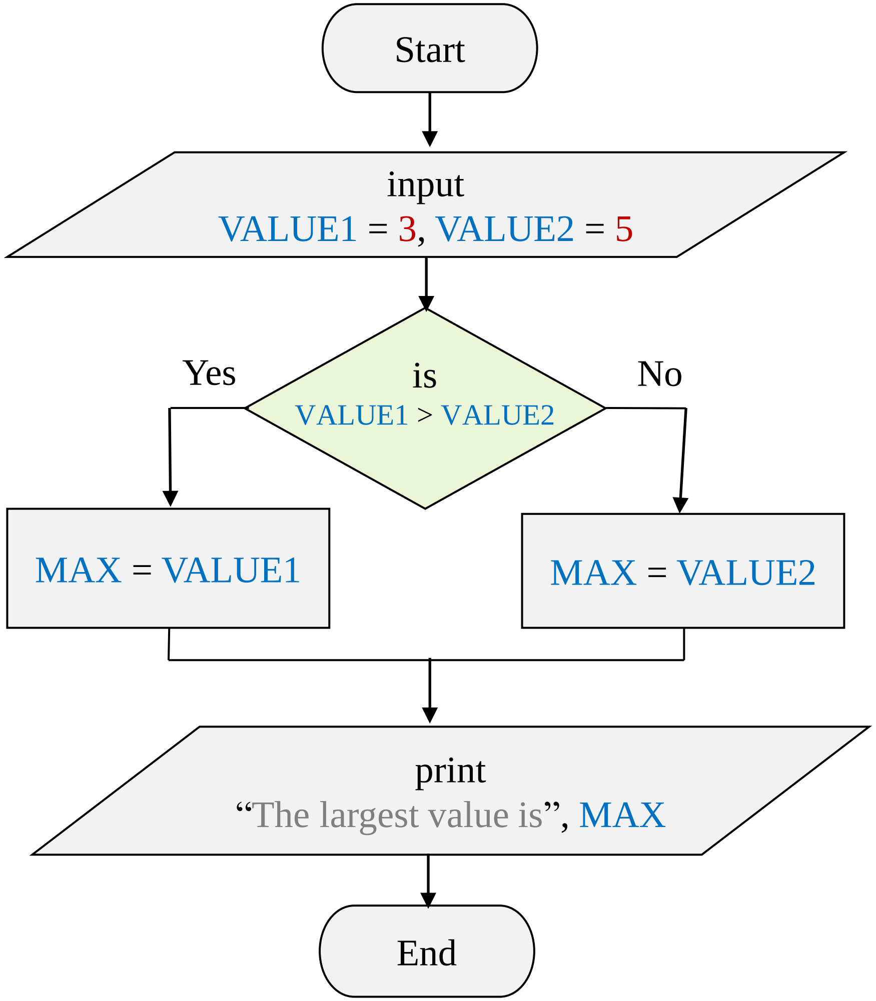
Determining The Largest Value

Verifying The Algorithm



Determining The Largest Value
Verifying The Algorithm

The Algorithm

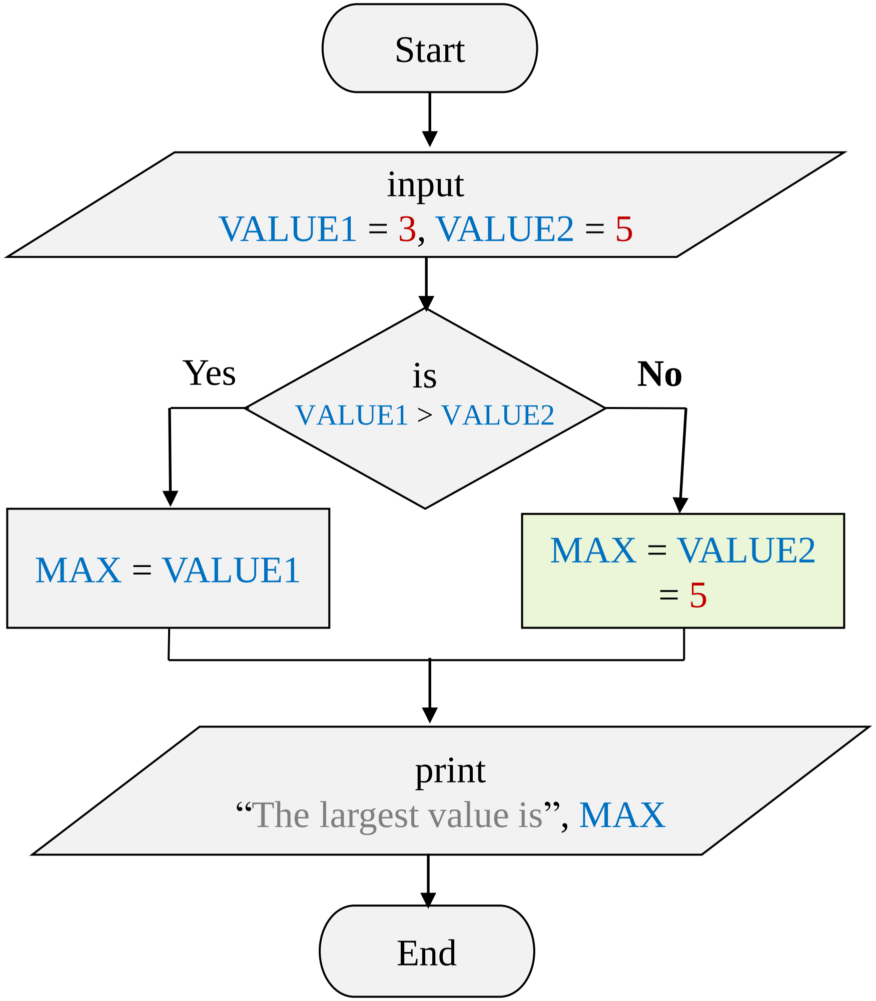


Test 1

The User

Determining The Largest Value
Verifying The Algorithm

The Algorithm

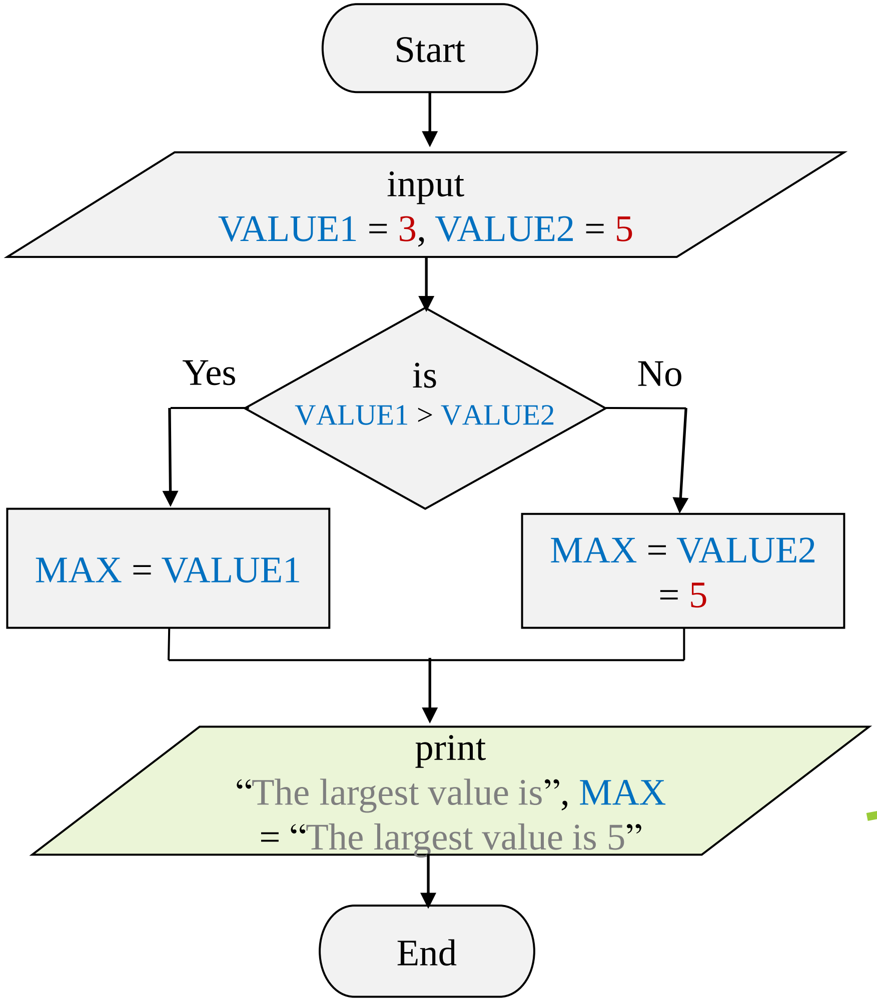


Test 1

The User

Determining The Largest Value
Verifying The Algorithm

The Algorithm



Start

input
VALUE1 = 3, VALUE2 = 5

is
VALUE1 > VALUE2

Yes

No

MAX = VALUE1

MAX = VALUE2
= 5

print
"The largest value is", MAX
= "The largest value is 5"

End

Test 1

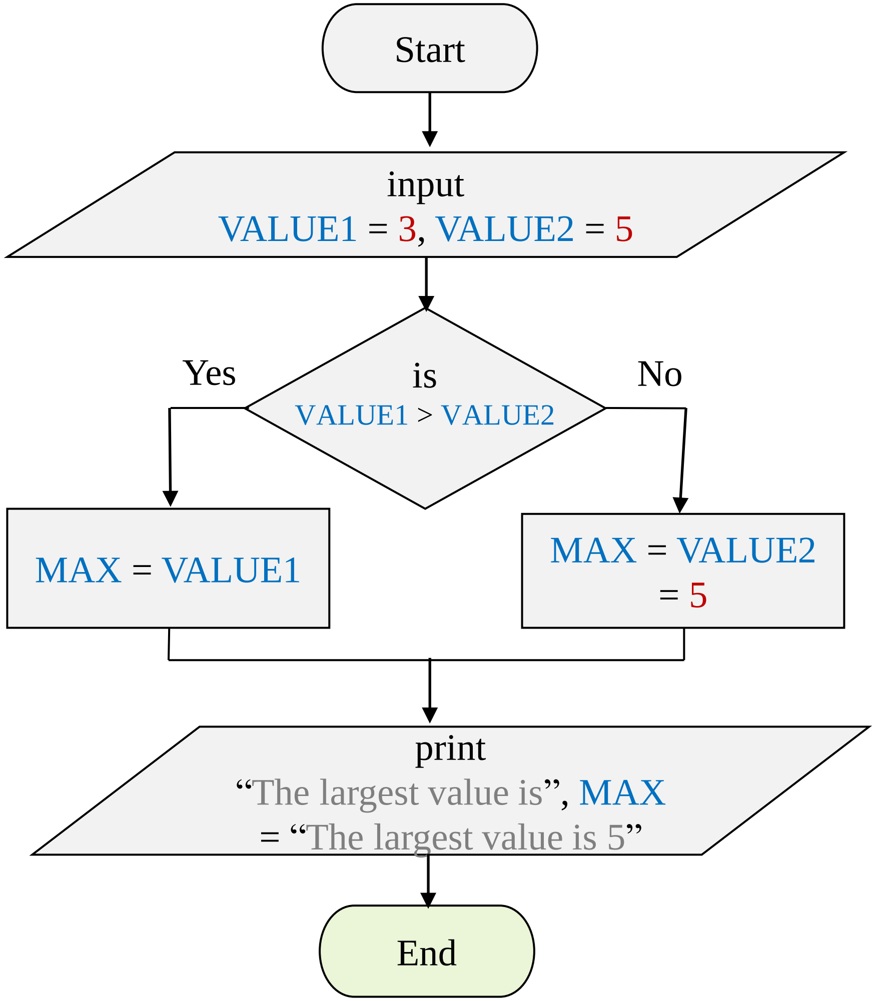
The User

"The largest value is 5"

Output:
The largest value is 5

Determining The Largest Value
Verifying The Algorithm

The Algorithm



Test 1

The User

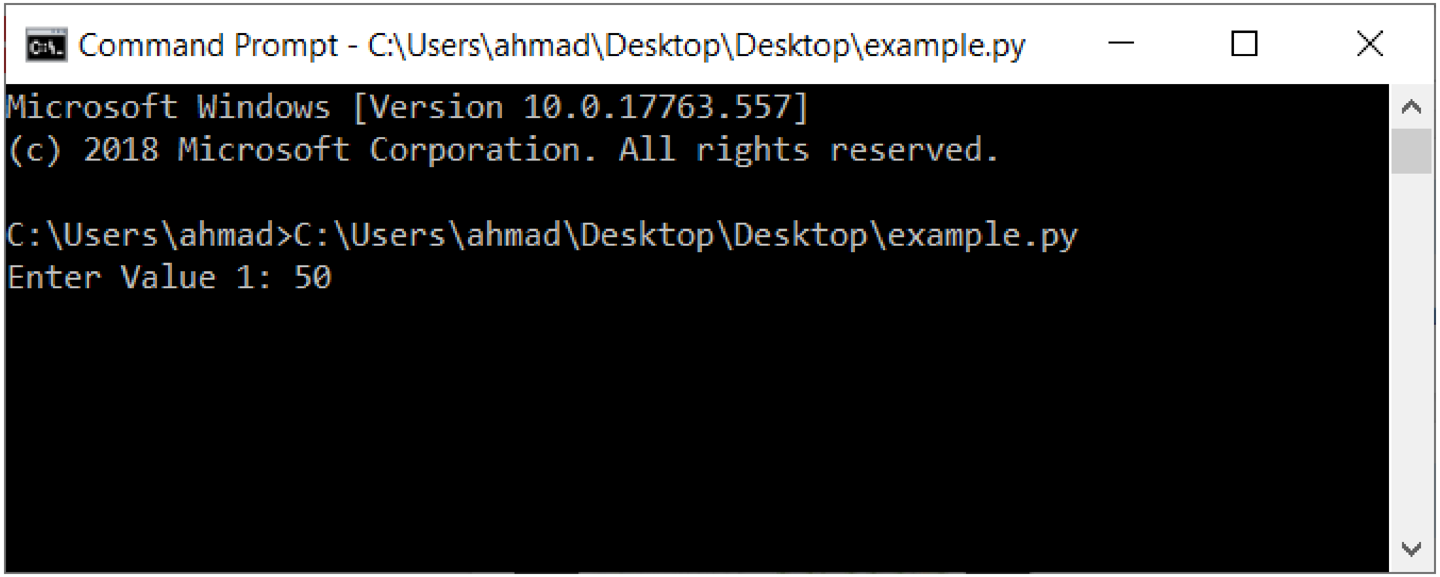
Output:
The largest value is 5

Determining The Largest Value

Verifying The Algorithm

```
Input VALUE1, VALUE2
if (VALUE1 > VALUE2) then
        MAX = VALUE1
else
        MAX = VALUE2
endif
print "The largest value is", MAX
```

Test 2

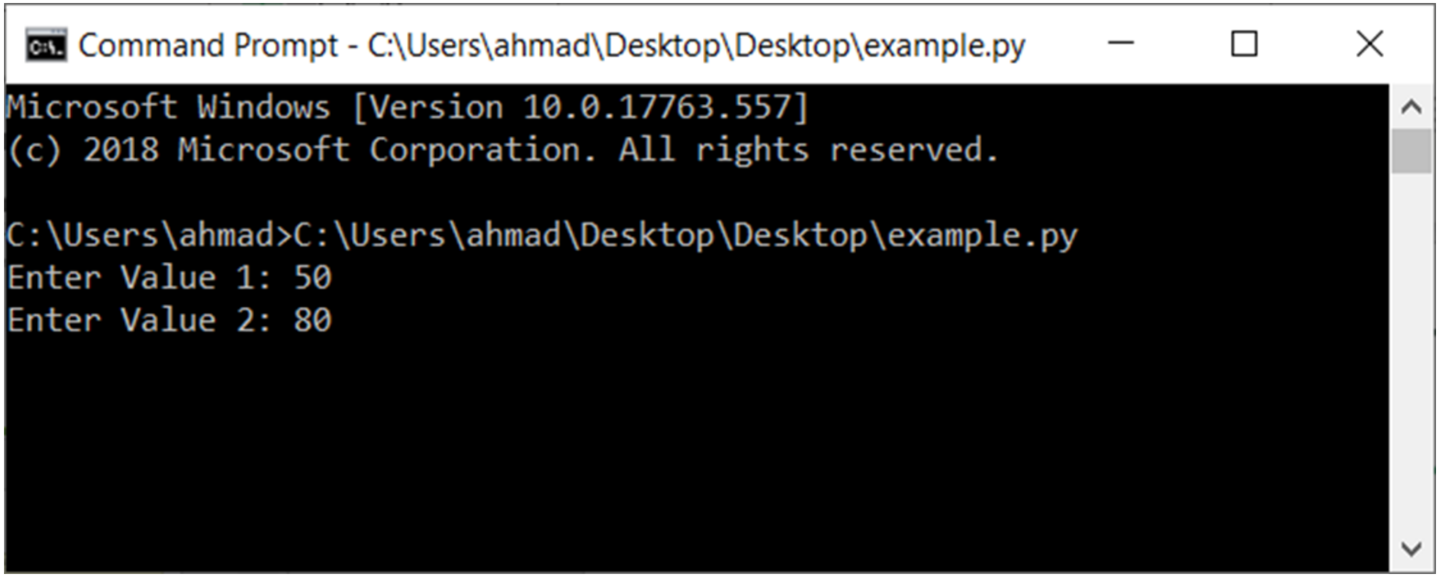


```
Microsoft Windows [Version 10.0.17763.557]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ahmad>C:\Users\ahmad\Desktop\Desktop\example.py
Enter Value 1: 50
```

Determining The Largest Value
Verifying The Algorithm

```
Input VALUE1, VALUE2
if (VALUE1 > VALUE2) then
    MAX = VALUE1
else
    MAX = VALUE2
endif
print “The largest value is”, MAX
```

Test 2



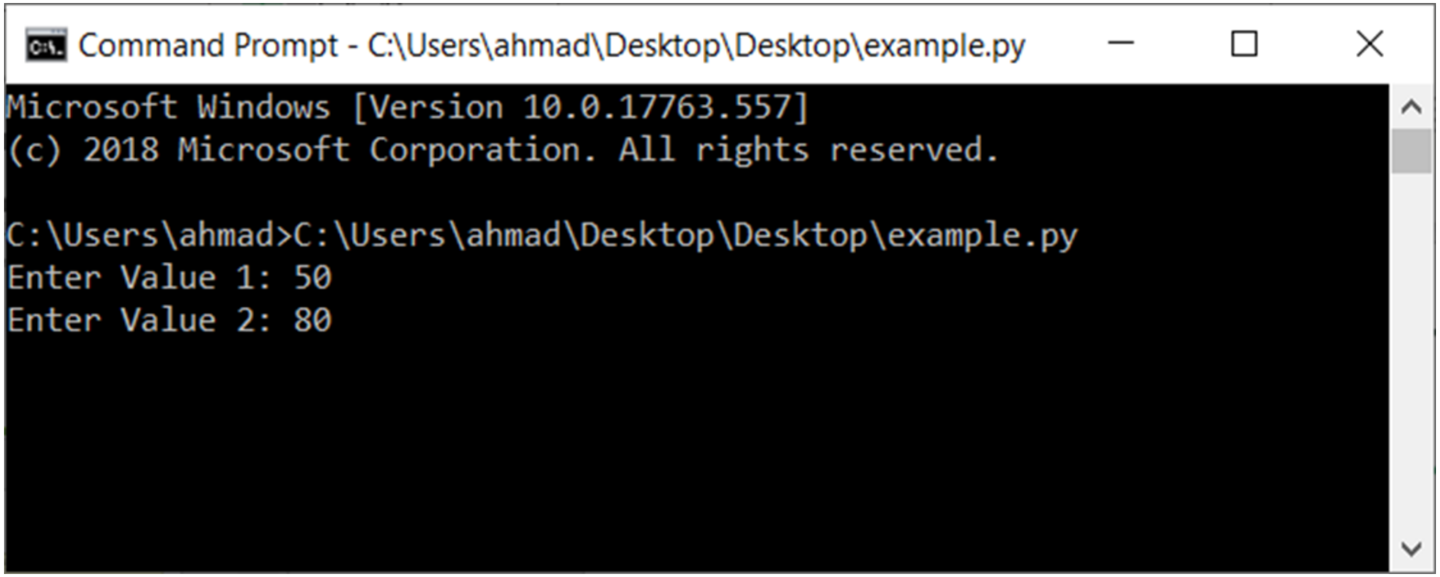
Command Prompt - C:\Users\ahmad\Desktop\Desktop\example.py

```
Microsoft Windows [Version 10.0.17763.557]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ahmad>C:\Users\ahmad\Desktop\Desktop\example.py
Enter Value 1: 50
Enter Value 2: 80
```

Determining The Largest Value
Verifying The Algorithm

```
Input VALUE1, VALUE2
if (VALUE1 > VALUE2) then
        MAX = VALUE1
else
        MAX = VALUE2
endif
print "The largest value is", MAX
```

Test 2



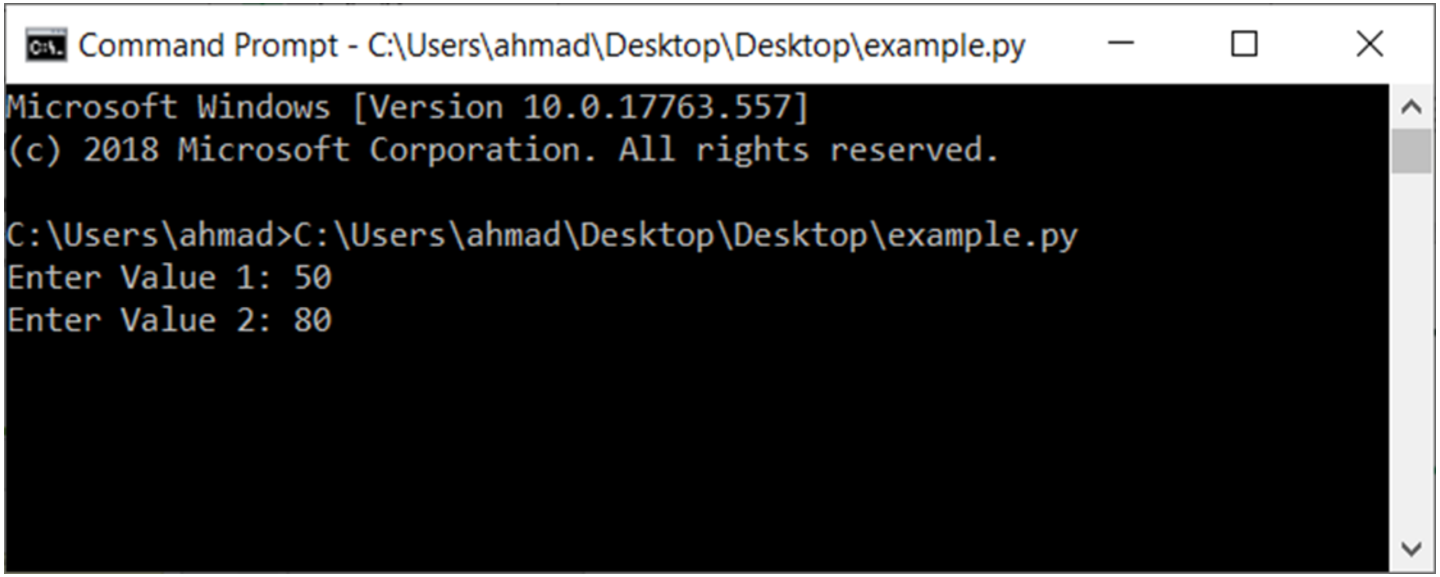
```
Microsoft Windows [Version 10.0.17763.557]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ahmad>C:\Users\ahmad\Desktop\Desktop\example.py
Enter Value 1: 50
Enter Value 2: 80
```

Determining The Largest Value

Verifying The Algorithm

```
Input VALUE1, VALUE2
if (VALUE1 > VALUE2) then
        MAX = VALUE1
else
        MAX = VALUE2
endif
print “The largest value is”, MAX
```

Test 2



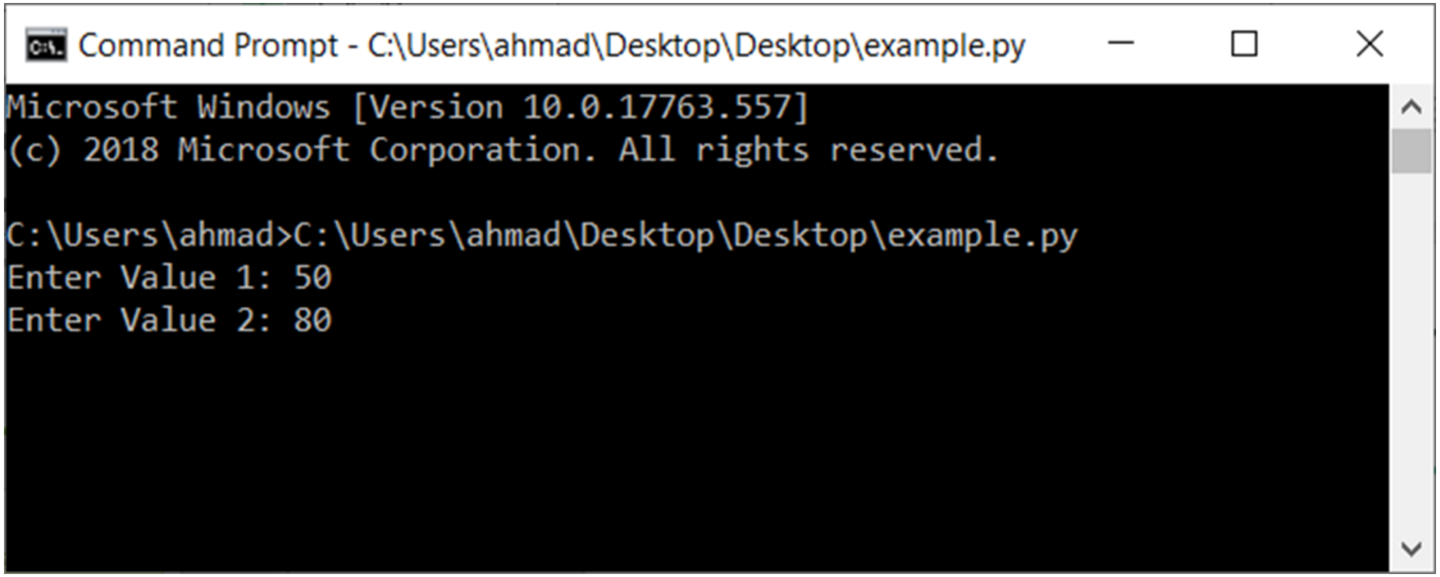
```
Microsoft Windows [Version 10.0.17763.557]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ahmad>C:\Users\ahmad\Desktop\Desktop\example.py
Enter Value 1: 50
Enter Value 2: 80
```

Determining The Largest Value

Verifying The Algorithm

```
Input VALUE1, VALUE2
if (VALUE1 > VALUE2) then
    MAX = VALUE1
else
    MAX = VALUE2
endif
print "The largest value is", MAX
```

Test 2



Command Prompt - C:\Users\ahmad\Desktop\Desktop\example.py

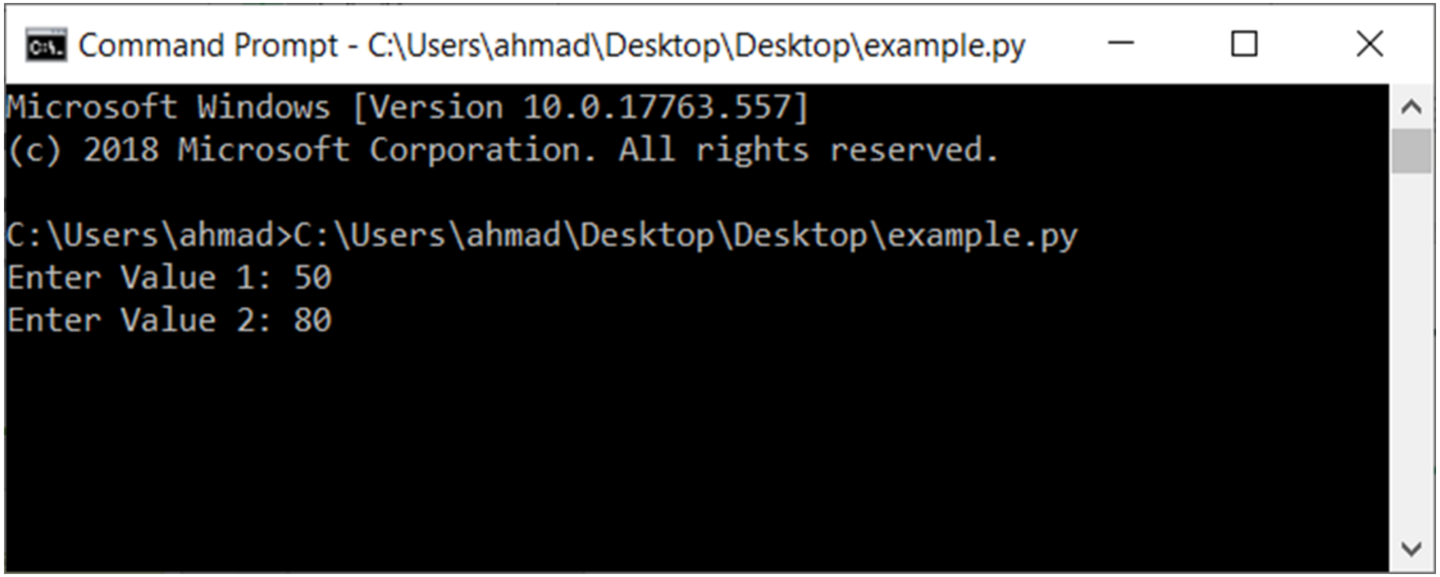
```
Microsoft Windows [Version 10.0.17763.557]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ahmad>C:\Users\ahmad\Desktop\Desktop\example.py
Enter Value 1: 50
Enter Value 2: 80
```

Determining The Largest Value

Verifying The Algorithm

```
Input VALUE1, VALUE2
if (VALUE1 > VALUE2) then
        MAX = VALUE1
else
        MAX = VALUE2
endif
print “The largest value is”, MAX
```

Test 2



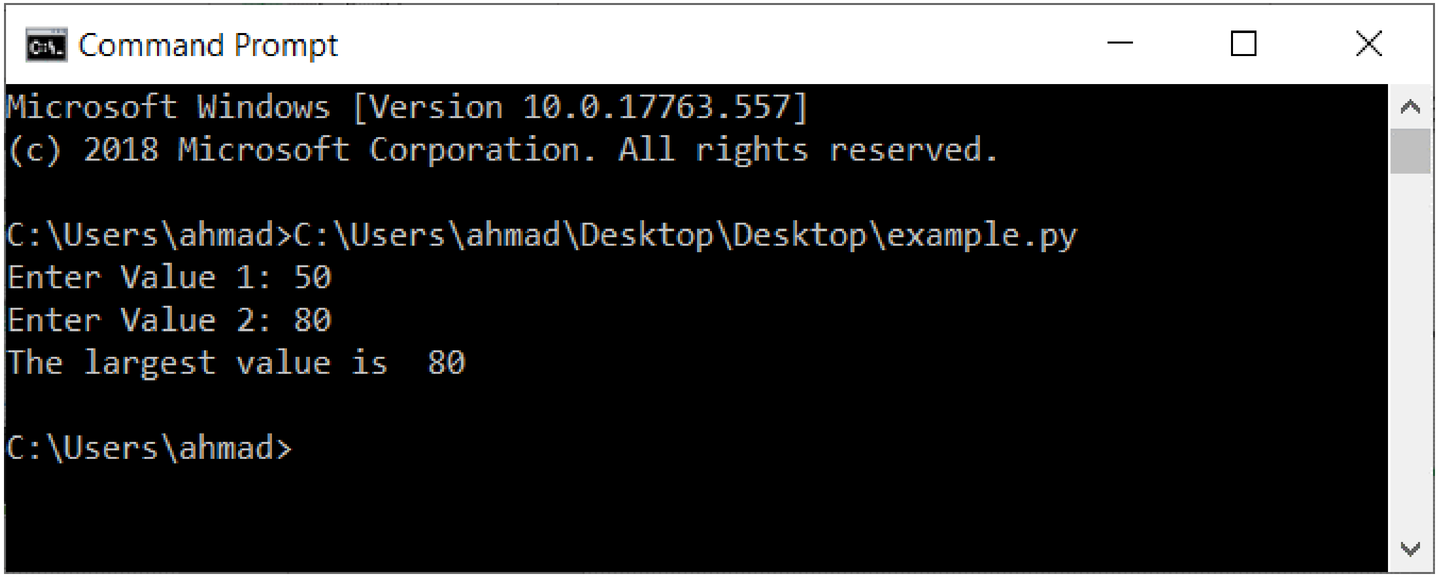
```
Microsoft Windows [Version 10.0.17763.557]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ahmad>C:\Users\ahmad\Desktop\Desktop\example.py
Enter Value 1: 50
Enter Value 2: 80
```

Determining The Largest Value

Verifying The Algorithm

```
Input VALUE1, VALUE2
if (VALUE1 > VALUE2) then
        MAX = VALUE1
else
        MAX = VALUE2
endif
print "The largest value is", MAX
```

Test 2



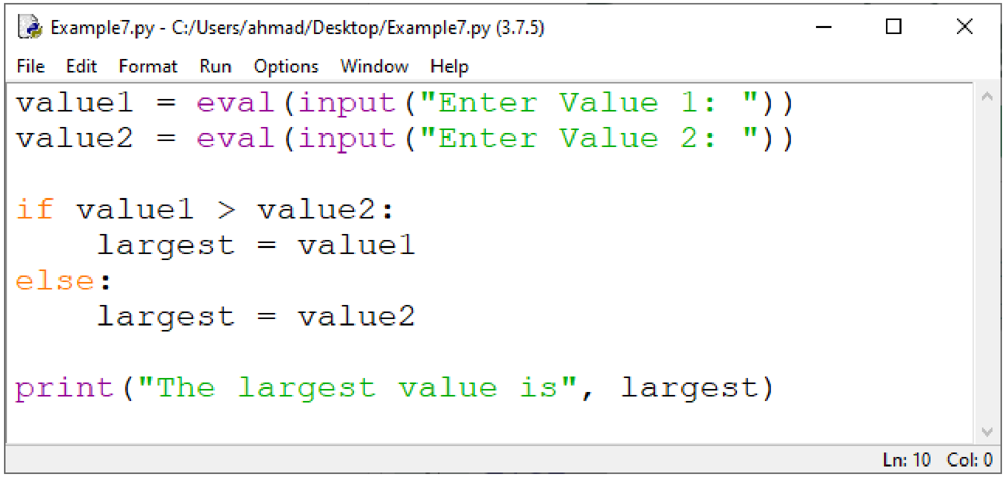
```
Microsoft Windows [Version 10.0.17763.557]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ahmad>C:\Users\ahmad\Desktop\Desktop\example.py
Enter Value 1: 50
Enter Value 2: 80
The largest value is  80

C:\Users\ahmad>
```

Determining The Largest Value
Python Code

Example7.py

```python
value1 = eval(input("Enter Value 1: "))
value2 = eval(input("Enter Value 2: "))

if value1 > value2:
    largest = value1
else:
    largest = value2

print("The largest value is", largest)
```

Run



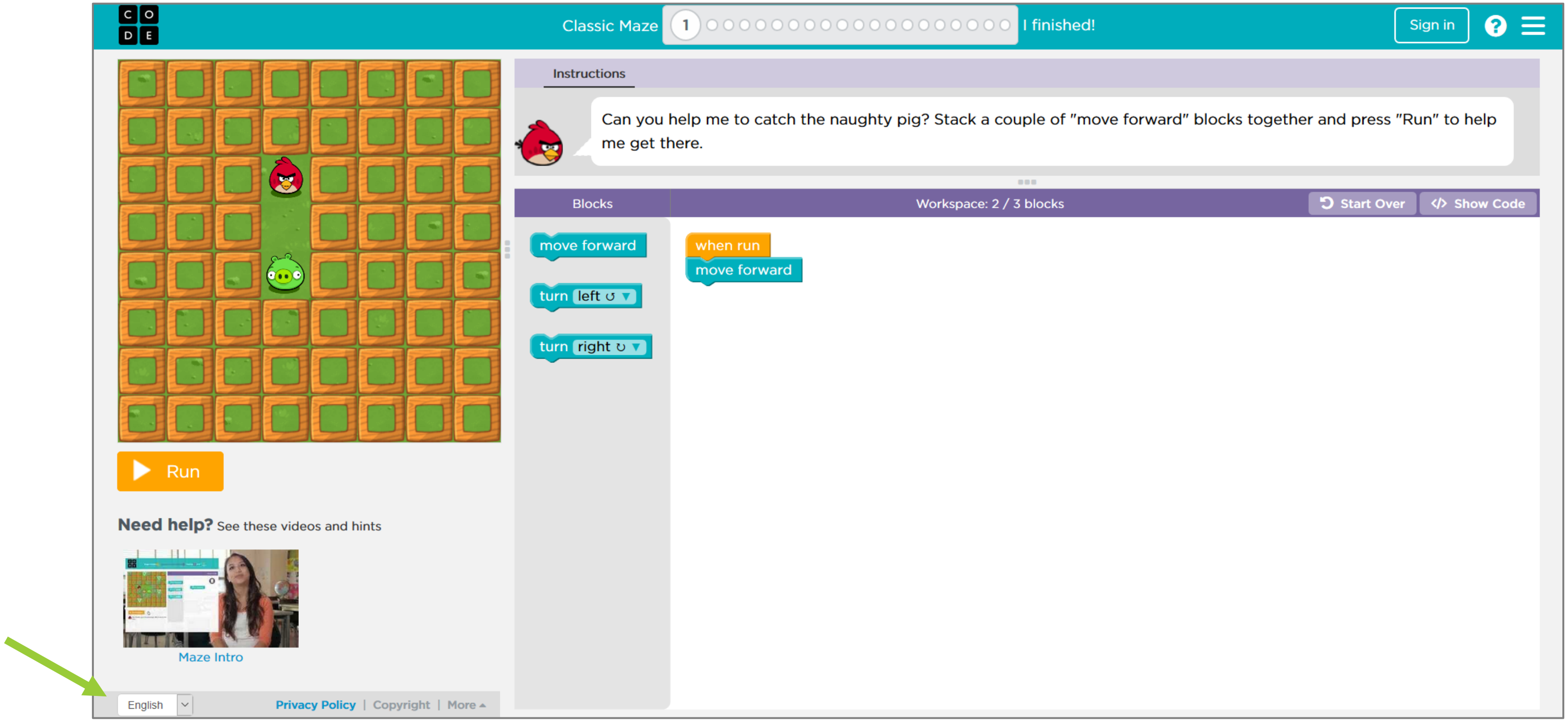
```python
value1 = eval(input("Enter Value 1: "))
value2 = eval(input("Enter Value 2: "))

if value1 > value2:
    largest = value1
else:
    largest = value2

print("The largest value is", largest)
```

End

- Play & Learn

Play & Learn

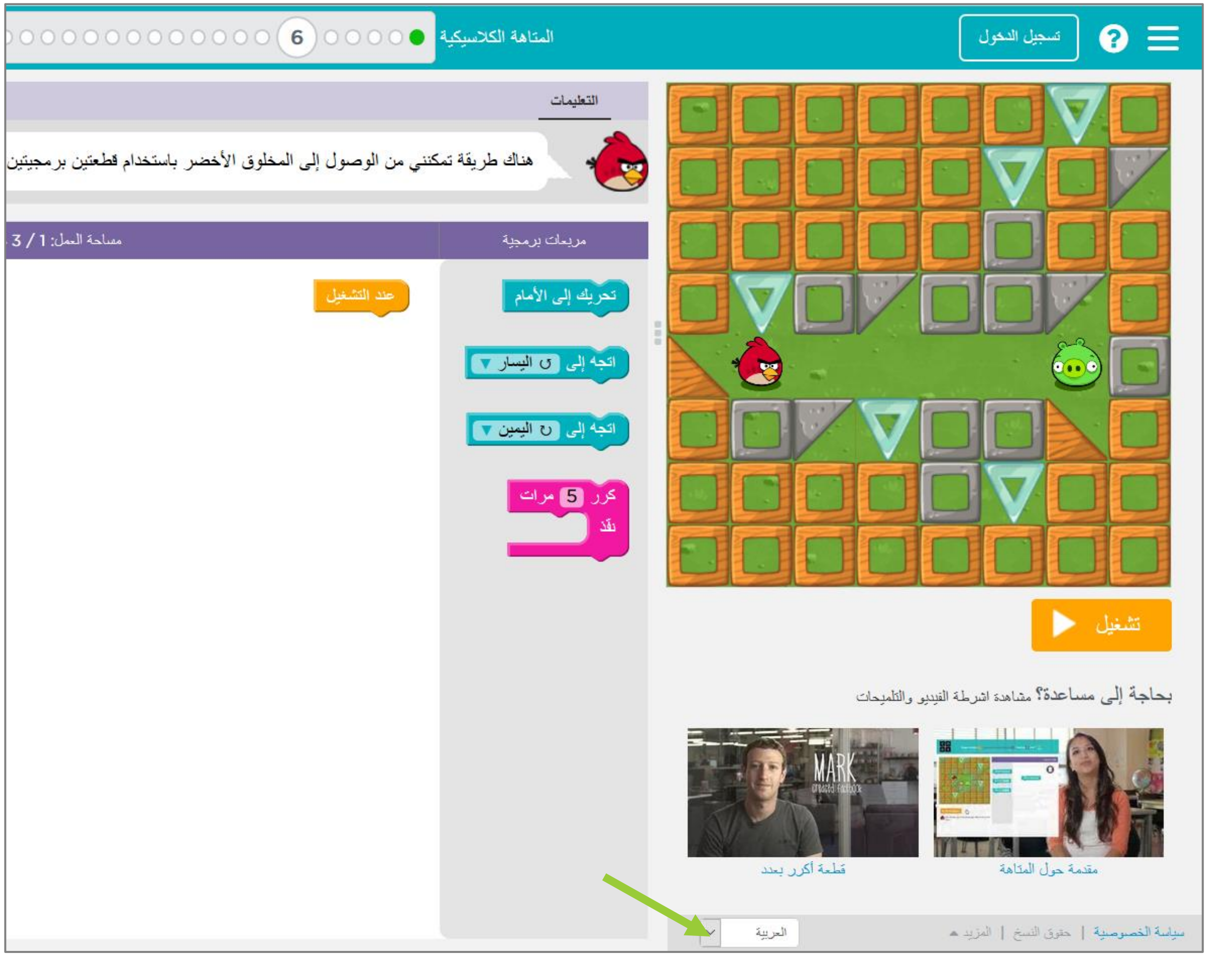
- Be familiar with basic logic and problem-solving techniques through practicing at Code.org.
- Visit https://studio.code.org/hoc/1 and play.



Can you help me to catch the naughty pig? Stack a couple of "move forward" blocks together and press "Run" to help me get there.

Play & Learn

- Note: You can select “Arabic” from the menu at the bottom.



هناك طريقة تمكنني من الوصول إلى المخلوق الأخضر باستخدام قطعتين برمجيتين

بحاجة إلى مساعدة؟ مشاهدة اشرطة الفيديو والتلميحات