# Function 2

Basic Programming Teaching Team 2022

# Objectives

**After studying this material, students should be able to:**

- Understand the concept of recursive functions
- Apply recursive functions to various problems

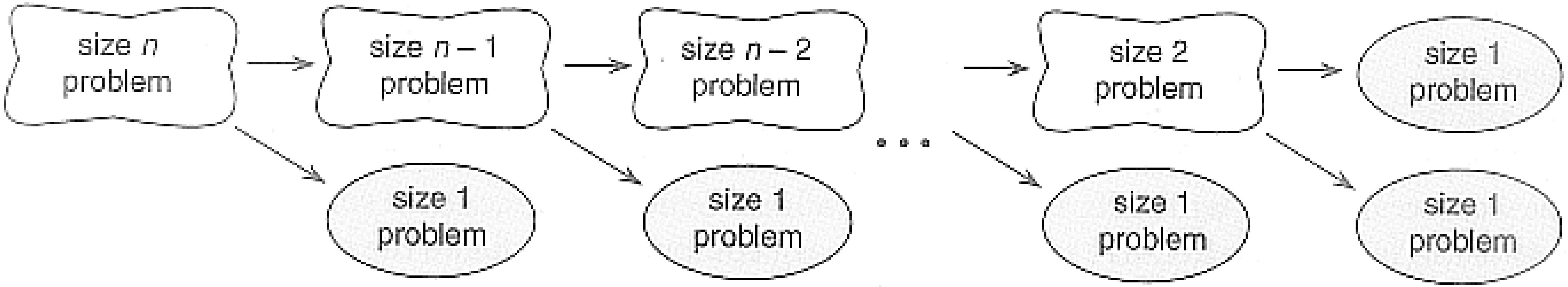
# Recursive Function

- Usually a function will be called by another function
- In a recursive function, a function contains a command **to call the function itself**. Thus, the function call process will occur repeatedly
- General form:

```
static ReturnDataType functionName(DataType parameterName){
    ...
    functionName(...)
    ...
}
```

# Recursive Function

- The problem solving strategy in recursive cases is called a decrease and conquer
- The idea is to reduce the size of the problem to a simple case that has a clear solution



- The recursive function will call itself, but the parameter values used for each call are different

# Recursive Function Components

- **Base Case**

  The recursion ends if the base case (limit value) is met

- **Recursion call / Reduction step**
  - The recursive function converges (approaches) towards the limit value
  - Usually has a **return** keyword to return the value to the function that called it

# Recursive Function Format

- In general, the recursive function format has the following form:

```
if (limit value)
   //solve the problem
else
   //redefines the problem using recursion
```

- The IF branch is the **base case**, while ELSE is the **recursion call**
- **Recursion calls provide the required repetition** to simplify problems and the **base case provides terminations**
- For recursion to stop, the recursion call must approach the base case in every recursive function call

# Recursive Function Tracing

The execution of the recursive function takes place in two stages:

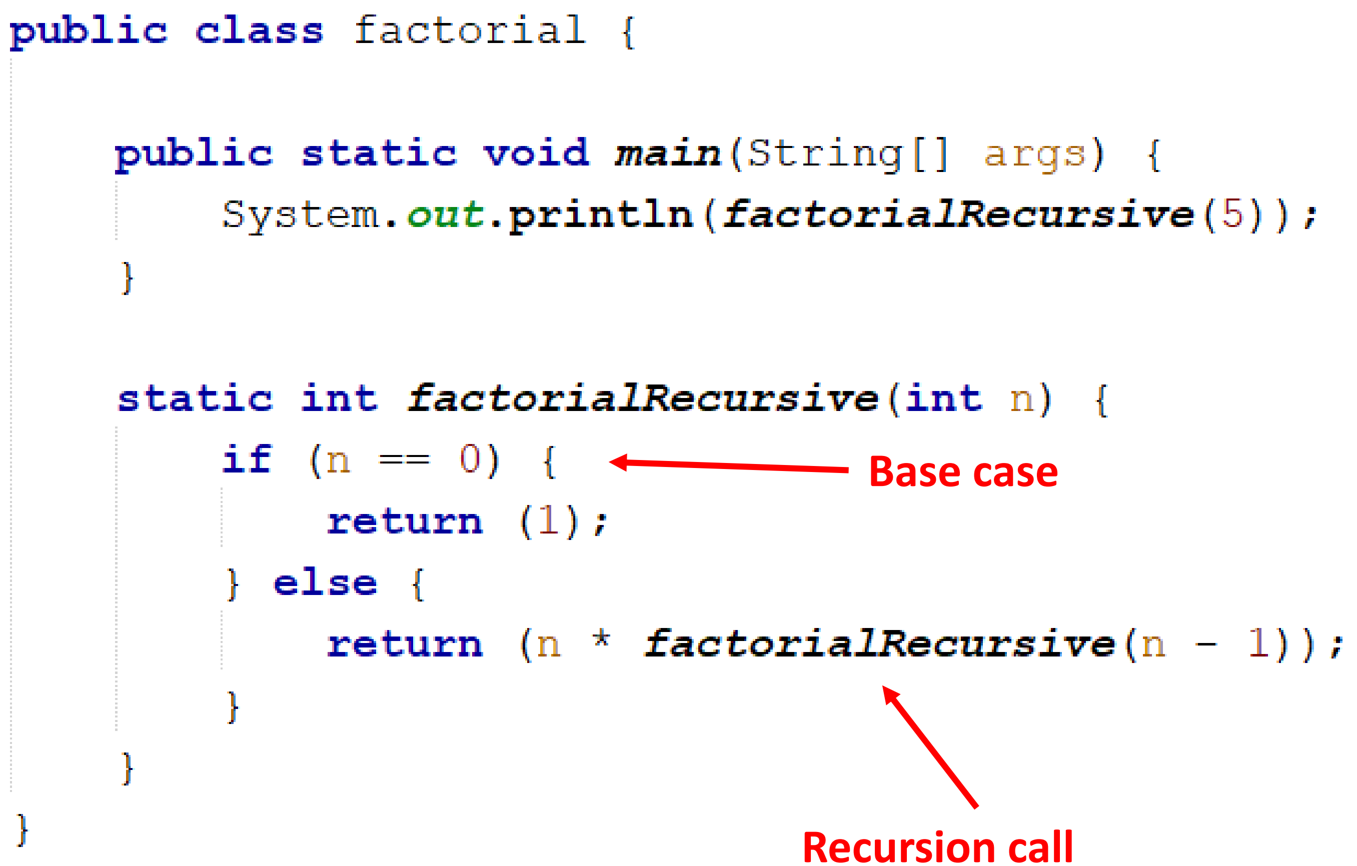
- **Expansion phase**: recursive function calls that get closer to the base case
- **Substitution phase**: the solution is calculated in reverse starting from the base case

# Example 1 Recursive Function

Factorial function

- **Base case**: n = 0
- **Recursion call**:

  f(n) = n * f(n-1)



```java
public class factorial {

    public static void main(String[] args) {
        System.out.println(factorialRecursive(5));
    }

    static int factorialRecursive(int n) {
        if (n == 0) {
            return (1);
        } else {
            return (n * factorialRecursive(n - 1));
        }
    }
}
```

Base case

Recursion call

# Example 1 Recursive Function - Tracing

**n * factorialRecursive(n-1)**

```
factorialRecursive(5)  = 5 * factorialRecursive(4)
                       = 5 * (4 * factorialRecursive(3))
                       = 5 * (4 * (3 * factorialRecursive(2)))
                       = 5 * (4 * (3 * (2 * factorialRecursive(1))))
                       = 5 * (4 * (3 * (2 * (1 * factorialRecursive(0)))))
```

**Expansion Phase**

```
                       = 5 * (4 * (3 * (2 * (1 * 1))))
                       = 5 * (4 * (3 * (2 * 1)))
                       = 5 * (4 * (3 * 2))
                       = 5 * (4 * 6)
                       = 5 * 24
                       = 120
```

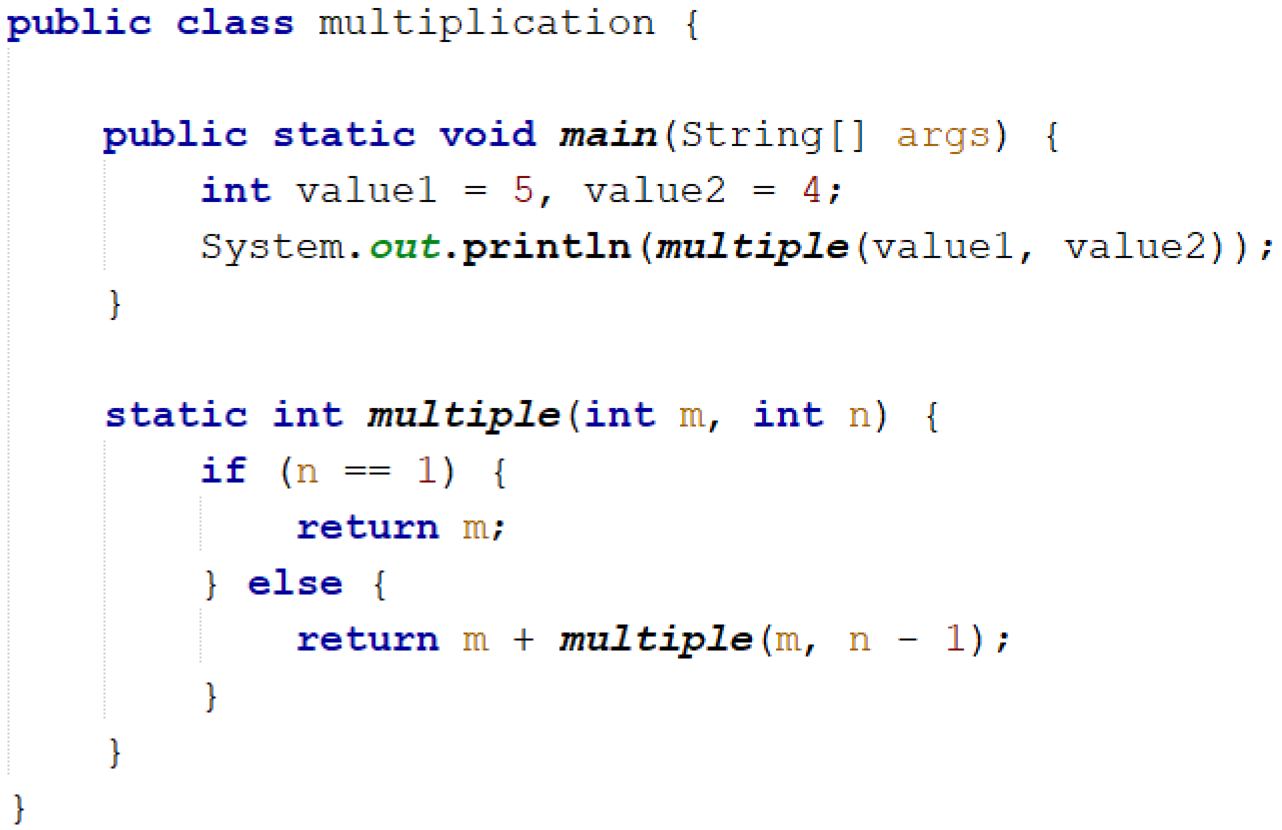
**Substitution Phase**

# Example 2 Recursive Function

- Suppose we want to create a recursive function to multiply integer m and integer n using addition
- We need to identify the base case and recursion call
  - **Base case**: if n is 1, the answer is m
  - **Recursion call**: m * n = m + m(n-1)

$$m * n \begin{cases} m, & n = 1 \\ m + m\,(n-1), & n>1 \end{cases}$$

# Example 2 Recursive Function - Tracing



```
public class multiplication {

    public static void main(String[] args) {
        int value1 = 5, value2 = 4;
        System.out.println(multiple(value1, value2));
    }

    static int multiple(int m, int n) {
        if (n == 1) {
            return m;
        } else {
            return m + multiple(m, n - 1);
        }
    }
}
```

**Expansion Phase**

```
multiple(5, 4) = 5 + multiple (5, 3)
               = 5 + (5 + multiple (5, 2))
               = 5 + (5 + (5+ multiple (5, 1)))
```

---

```
               = 5 + (5 + (5 + 5))
               = 5 + (5 + 10)
               = 5 + 15
               = 20
```

**Substitution Phase**

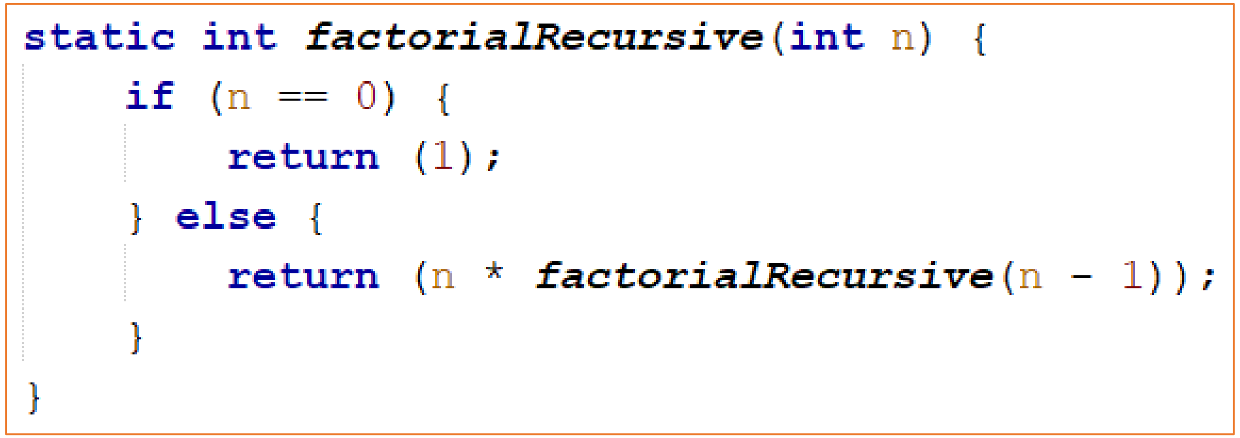
# Recursive Function Vs Iterative Function

# Recursive Function Vs Iterative Function

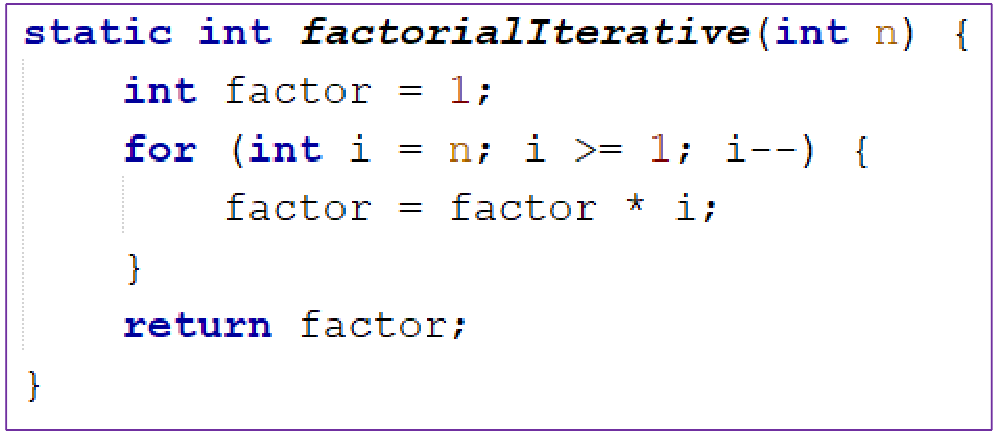
- Loops with a selection structure (IF-ELSE), and a function call itself
- The loop stops when the base case is fulfilled
- Endless repetition if the base case is never fulfilled
- Requires more memory and higher processor work because it calls many functions
- It reads more clearly, the model is closer to the problem, example: factorial

- Loops with repetition structure (FOR / WHILE)
- The loop stops when the condition is FALSE
- Repeating without stopping if the loop condition is always correct
- Requires less memory and lower processor work because the repetition process is in one function
- It reads less clearly, the model is not close to the problem

# Recursive Function Vs Iterative Function

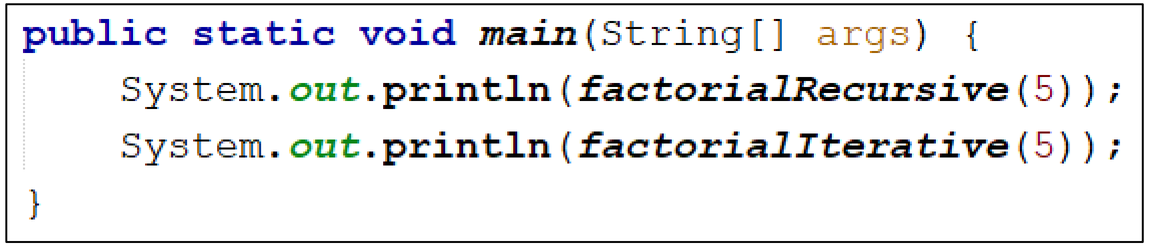


```java
static int factorialRecursive(int n) {
    if (n == 0) {
        return (1);
    } else {
        return (n * factorialRecursive(n - 1));
    }
}
```



```java
static int factorialIterative(int n) {
    int factor = 1;
    for (int i = n; i >= 1; i--) {
        factor = factor * i;
    }
    return factor;
}
```

Main function



```java
public static void main(String[] args) {
    System.out.println(factorialRecursive(5));
    System.out.println(factorialIterative(5));
}
```

# When Do We Use Recursive?

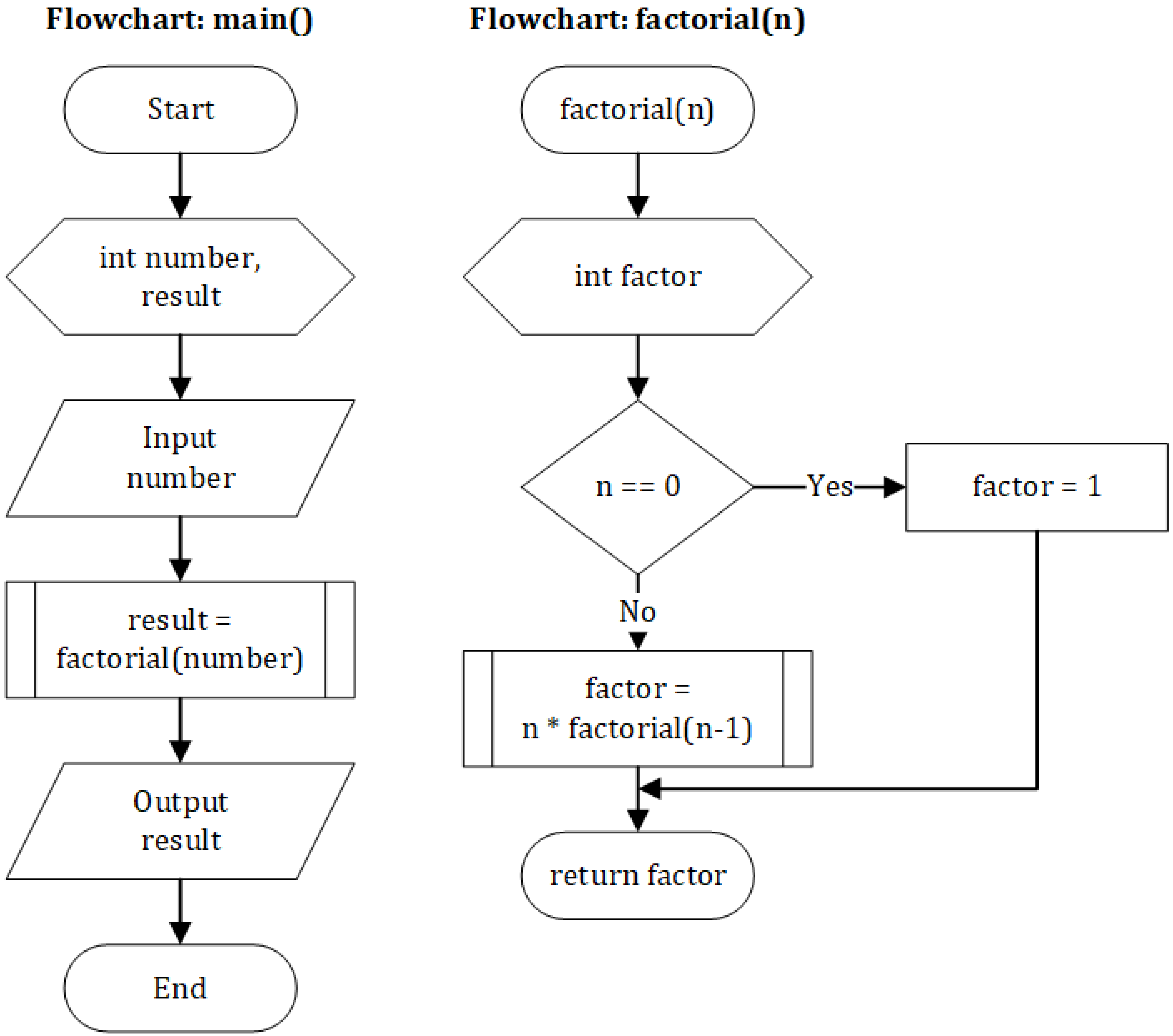
When:

- Problem solving is difficult to do iteratively
- Does not consider the memory saving factor and program execution speed

# Example 1

Create a flowchart to calculate the factorial of a number using a recursive function!

# Example 1 - Answer



# Example 2

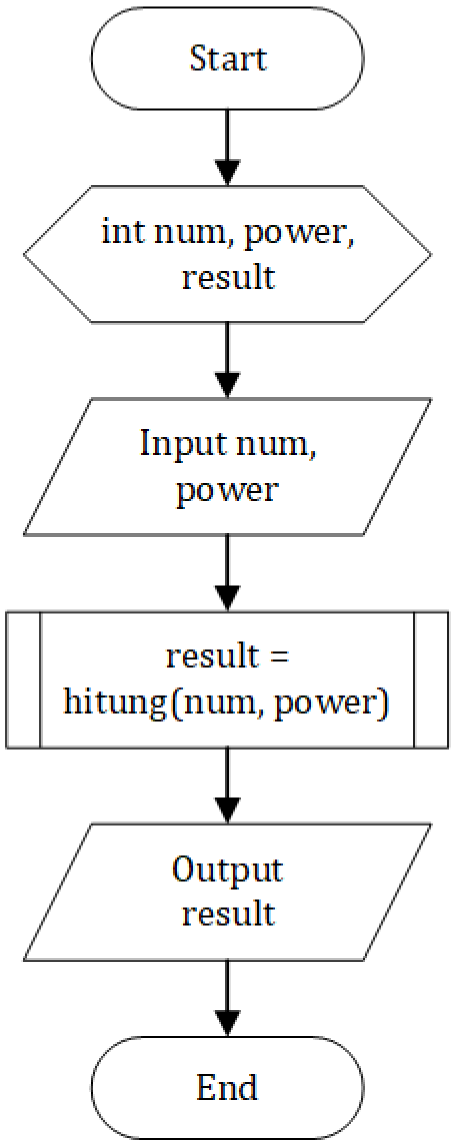
There is a program to calculate the value of X to the power of Y. As we know, the value of X to the power of Y is calculated by X times X (Y- 1) times, but if Y is 0 (X to the power of 0) then the value of X is 1.

So to calculate the value of X to the power of Y, the program must provide a limit that if Y = 0 then the value of X becomes 1.

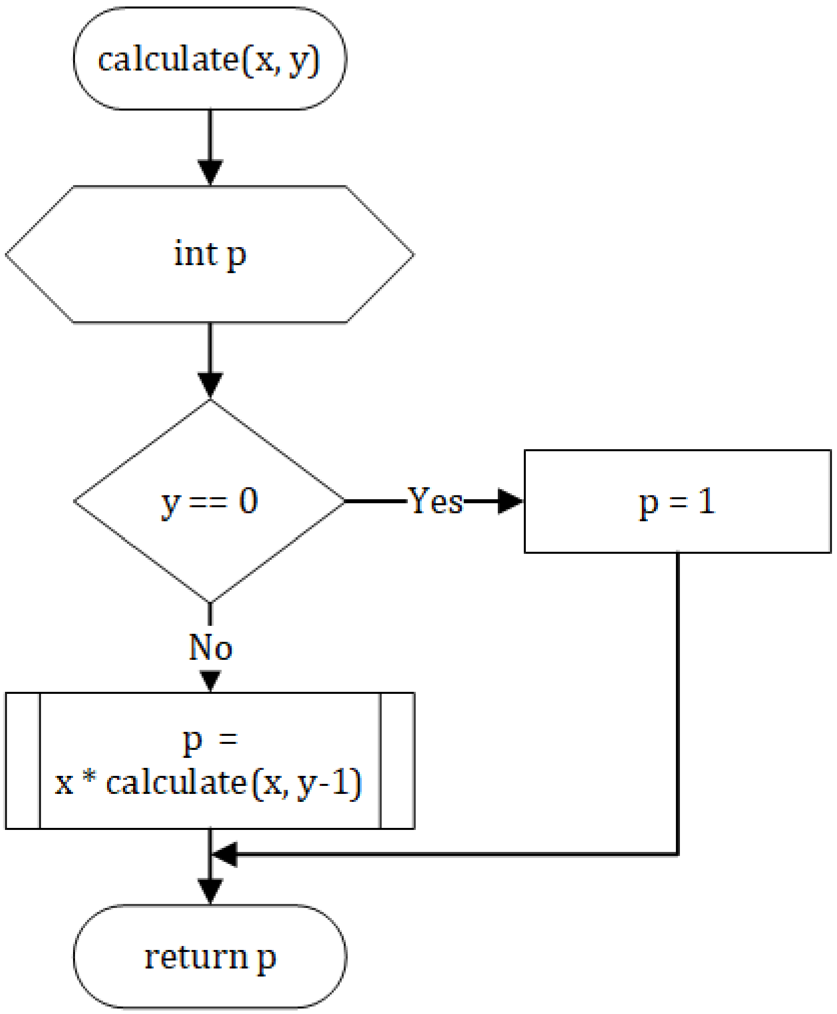
Create the flowchart!

# Example 2 - Answer

**Flowchart: main()**



**Flowchart: calculate(x, y)**



# Assignment

1. Create a flowchart to calculate a Factorial number. Suppose that the factorial number is 5!, then calculate the result of 1 * 2 * 3 * 4 * 5.
2. Calculate the return on someone's investment on the purchase of a company's stock. The profit obtained is based on the annual interest rate of 5.5%. Create a flowchart to determine the amount of money after a few years, for example 20 years!