

## OVERVIEW

The following are the coverage for Recursion:

- What is Recursion?
- **Recursive Functions: Examples**
- Recursive Functions: Returning Value
- Recursive Functions: Call by Reference
- Recursion in Arrays
- How to Design Recursive Functions

Content Copyright Nanyang Technological University

2

There are 6 main sections to cover for Recursion as shown. This video lesson focuses on the second part where we will look into some examples of recursive functions.

## LESSON OBJECTIVES

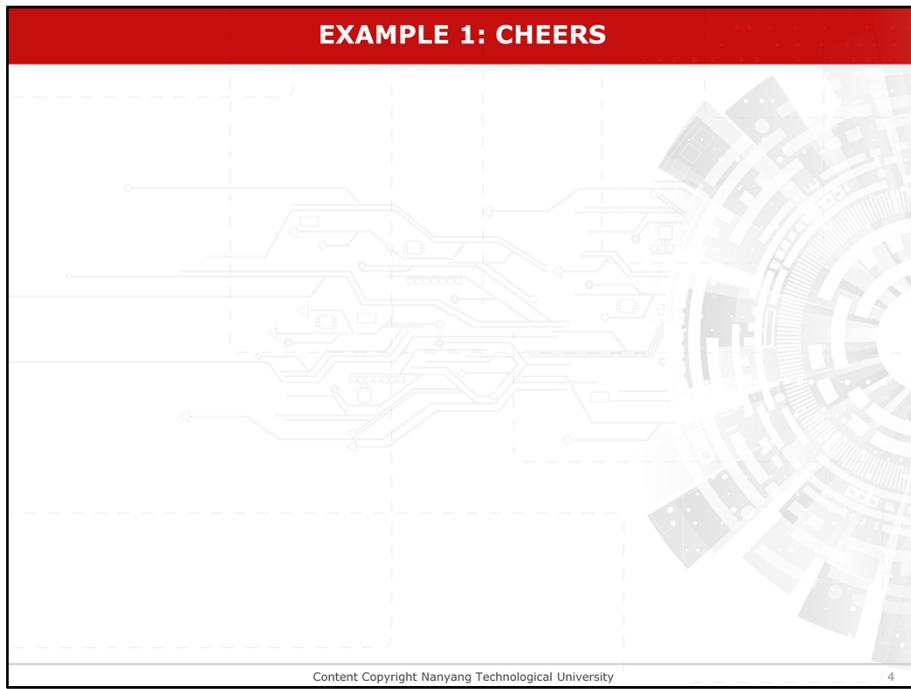
After this lesson, you should be able to:

- Explain recursion and how it works with some examples

Content Copyright Nanyang Technological University

3

After this lesson, you should be able to explain recursion and how it works with some examples.



Example 1: Cheers

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

Content Copyright Nanyang Technological University

5

The recursive function called **cheers()** is implemented as shown.

## EXAMPLE 1: CHEERS

```
void cheers( int n )
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

The function **cheers()** accepts an integer parameter **n**.

Content Copyright Nanyang Technological University

6

The function **cheers()** accepts an integer parameter **n**.

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

Content Copyright Nanyang Technological University

7

There are two parts of coding in the function.

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

terminating condition ( $n \leq 1$ )

The first part is the code that handles the case when the **terminating condition** occurs.

Content Copyright Nanyang Technological University

8

The first part is the code that handles the case when the terminating condition occurs.

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

terminating condition ( $n \leq 1$ )

The first part is the code that handles the case when the **terminating condition** occurs. This will happen when the parameter  $n \leq 1$ .

Content Copyright Nanyang Technological University

9

This will happen when the parameter  $n$  is lesser than or equal to 1.

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

terminating condition ( $n \leq 1$ )

The first part is the code that handles the case when the **terminating condition** occurs. This will happen when the parameter  $n \leq 1$ .

Content Copyright Nanyang Technological University

10

In this case, the character string "Hurrah" will be printed on the screen.

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

recursive condition ( $n > 1$ )

The second part is the code that handles the case when the **recursive condition** occurs.

Content Copyright Nanyang Technological University

11

The second part is the code that handles the case when the recursive condition occurs.

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

recursive condition ( $n > 1$ )

The second part is the code that handles the case when the **recursive condition** occurs. This will happen when  $n$  is greater than 1.

Content Copyright Nanyang Technological University

12

This will happen when  $n$  is greater than 1.

## EXAMPLE 1: CHEERS

```
void cheers( int n )
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

recursive condition ( $n > 1$ )

The second part is the code that handles the case when the **recursive condition** occurs. This will happen when  $n$  is greater than 1.

Content Copyright Nanyang Technological University

13

In this case, the character string "Hip" will be printed on the screen.

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

recursive condition ( $n > 1$ )

And the function cheers() will be called again with the parameter n reduced by 1.

Content Copyright Nanyang Technological University

14

And the function cheers() will be called again with the parameter n reduced by 1.

## EXAMPLE 1: CHEERS

```
void cheers( int n )
{
    if (n <= 1)           ← terminating condition (n <= 1)
        printf("Hurrah \n");
    else                  ← recursive condition (n > 1)
    {
        printf("Hip \n");
        cheers(n-1);      ← recursive function with updated
                            parameter getting closer to
                            terminating condition
    }
}
```

Note that the recursive condition is implemented to ensure that the recursive function will be called again with the updated parameter getting closer to the terminating condition.

Content Copyright Nanyang Technological University

15

Note that the recursive condition is implemented to ensure that the recursive function will be called again with the updated parameter getting closer to the terminating condition.

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

What does the recursive function print  
when called with **cheers(4)**?

Content Copyright Nanyang Technological University

16

The program tracing for **cheers (4)** is illustrated as follows.

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

n=4  
cheers(4)

Content Copyright Nanyang Technological University

17

When the parameter **n** is equal to 4,

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

n=4  
cheers(4)

Content Copyright Nanyang Technological University

18

the code for recursive condition is then executed.

## EXAMPLE 1: CHEERS

The diagram illustrates the execution of a recursive function. On the left, the code for the `cheers` function is shown:

```
void cheers( int n )
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

In the middle, a call to `cheers(4)` is highlighted. To its right, a callout shows the result of the recursive step: `printf("Hip \n")`. At the bottom left, the output is shown as "Hip".

n=4      cheers(4)      printf("Hip \n")

Output      Hip

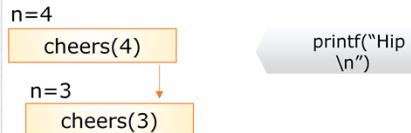
Content Copyright Nanyang Technological University      19

The string "**Hip**" will be printed on the screen,

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

Output  
Hip



Content Copyright Nanyang Technological University

20

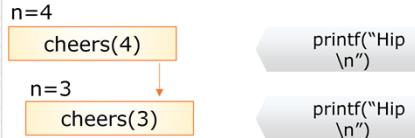
and the function **cheers (3)** will be called with **n** is equal to 3.

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

### Output

```
Hip  
Hip
```



21

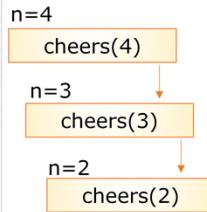
This will be repeated to print the string "Hip" for two more times until the function **cheers (1)** is called with **n** reduced to 1.

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

### Output

Hip  
Hip



printf("Hip \n")  
printf("Hip \n")

Content Copyright Nanyang Technological University

22

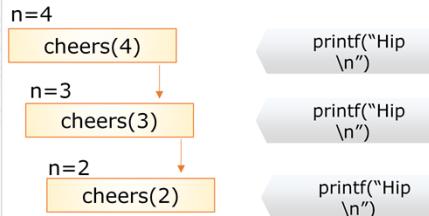
[no audio]

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

### Output

Hip  
Hip  
Hip



Content Copyright Nanyang Technological University

23

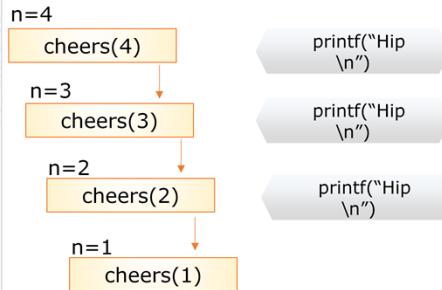
[no audio]

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

### Output

Hip  
Hip  
Hip



Content Copyright Nanyang Technological University

24

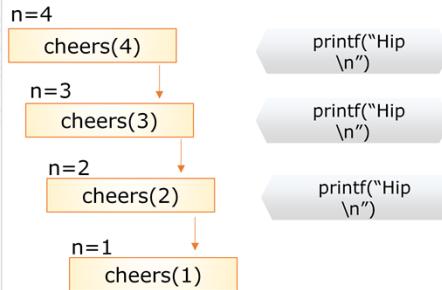
[no audio]

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

### Output

Hip  
Hip  
Hip



Content Copyright Nanyang Technological University

25

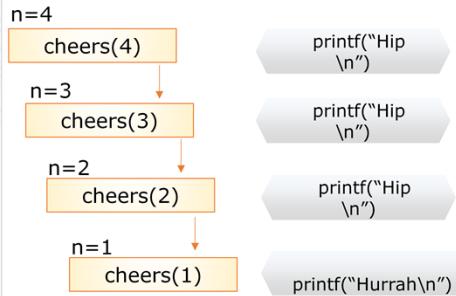
When  $n$  is equal to 1.

## EXAMPLE 1: CHEERS

```
void cheers( int n)
{
    if (n <= 1)
        printf("Hurrah \n");
    else
    {
        printf("Hip \n");
        cheers(n-1);
    }
}
```

**Output**

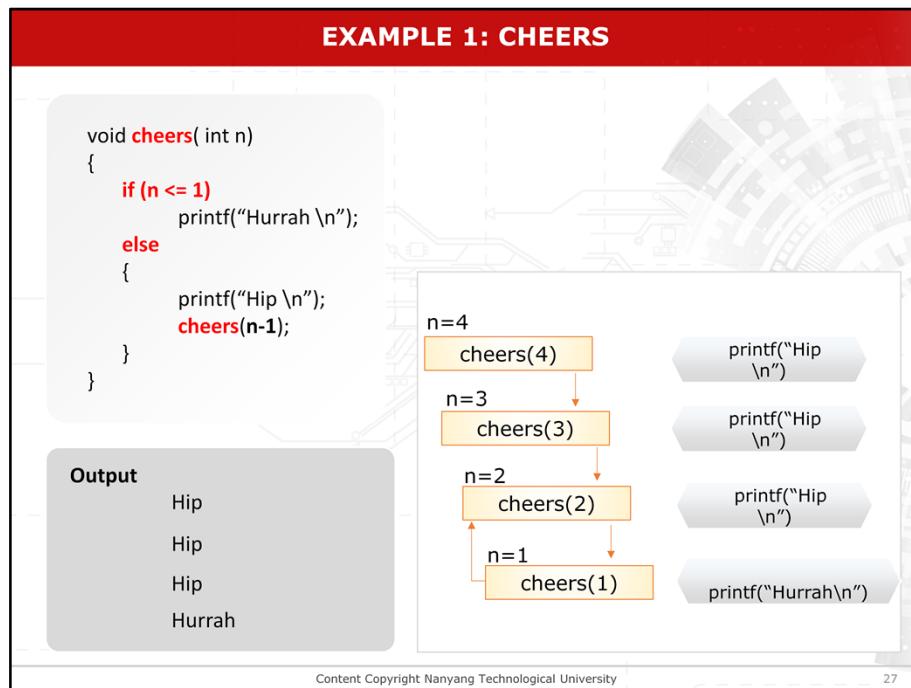
```
Hip
Hip
Hip
Hurrah
```



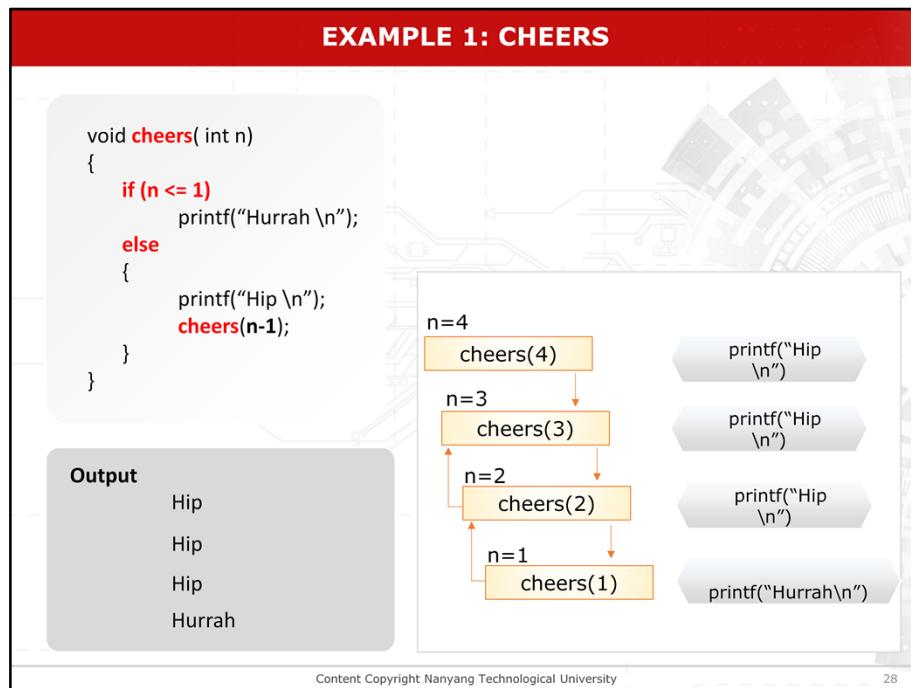
Content Copyright Nanyang Technological University

26

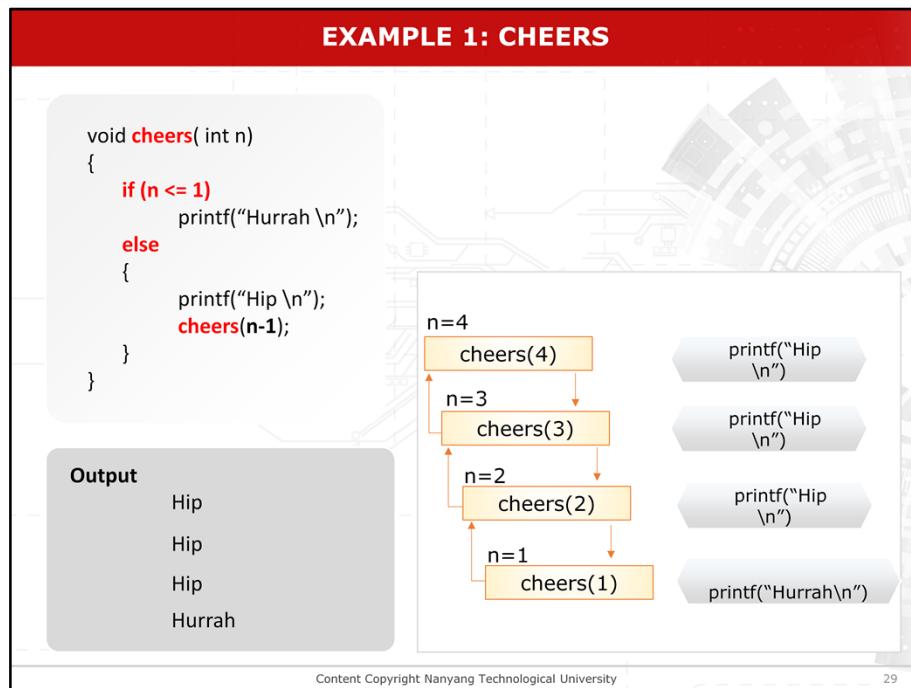
the string "**Hurrah**" will be printed on the screen.



When `cheers(1)` has completed execution, the control is returned to the previous function call executing `cheers(2)`.

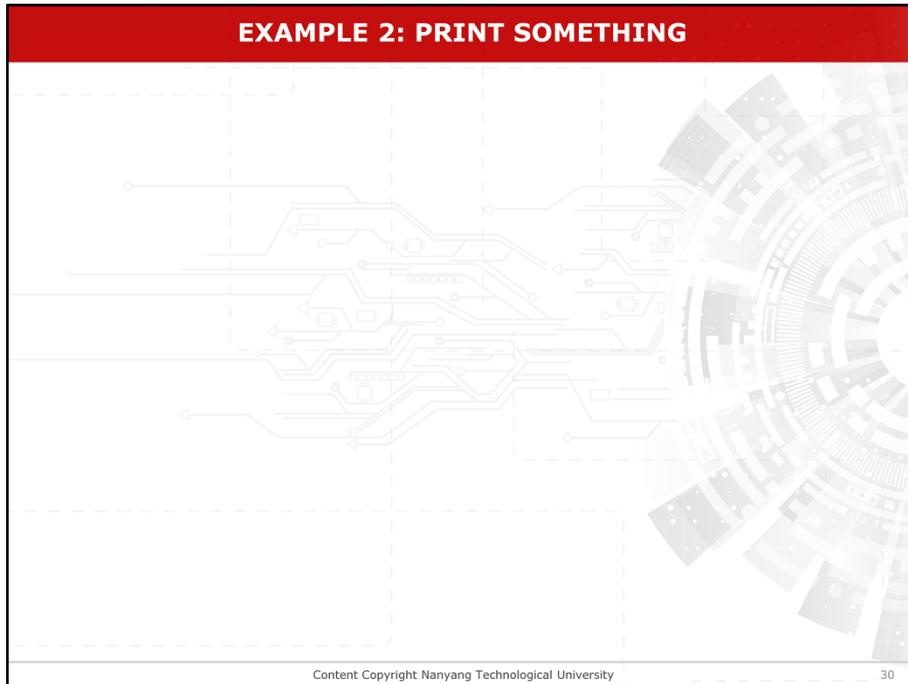


It will in turn complete the execution, and the control returns to the previous calling function **cheers(3)**.



The program stops execution when **cheers(4)** has completed the execution.

## EXAMPLE 2: PRINT SOMETHING



Content Copyright Nanyang Technological University

30

Example 2: Print Something

## EXAMPLE 2: PRINT SOMETHING

```
void printSomething( int n )
{
    if (n <= 0)
        return;
    else {
        printf("*");
        printSomething(n-1);
        printf("!");
    }
}
```

Content Copyright Nanyang Technological University

31

The recursive function called **print Something()** is implemented as shown.

## EXAMPLE 2: PRINT SOMETHING

```
void printSomething( int n )
{
    if (n <= 0)
        return;
    else {
        printf("*");
        printSomething(n-1);
        printf("!");
    }
}
```

The function **printSomething()** accepts an integer parameter **n**.

Content Copyright Nanyang Technological University

32

The function **print Something()** accepts an integer parameter **n**.

## EXAMPLE 2: PRINT SOMETHING

```
void printSomething( int n )
{
    if (n <= 0)
        return;
    else {
        printf("*");
        printSomething(n-1);
        printf("!");
    }
}
```

Content Copyright Nanyang Technological University

33

There are two parts of coding in the function.

## EXAMPLE 2: PRINT SOMETHING

```
void printSomething( int n )
{
    if (n <= 0)
        return;
    else {
        printf("*");
        printSomething(n-1);
        printf("!");
    }
}
```

terminating condition

The first part is the code that handles the case when the **terminating condition** occurs.

Content Copyright Nanyang Technological University

34

The first part is the code that handles the case when the terminating condition occurs.

## EXAMPLE 2: PRINT SOMETHING

```
void printSomething( int n )
{
    if (n <= 0)
        return;
    else {
        printf("*");
        printSomething(n-1);
        printf("!");
    }
}
```

terminating condition ( $n \leq 0$ )

The first part is the code that handles the case when the **terminating condition** occurs. This will happen when the parameter  $n \leq 0$ .

Content Copyright Nanyang Technological University

35

This will happen when the parameter  $n$  is lesser than or equal to 0.

## EXAMPLE 2: PRINT SOMETHING

```
void printSomething( int n )
{
    if (n <= 0)
        return;
    else {
        printf("*");
        printSomething(n-1);
        printf("!");
    }
}
```

terminating condition ( $n \leq 0$ )

The first part is the code that handles the case when the **terminating condition** occurs. This will happen when the parameter  $n \leq 0$ .

Content Copyright Nanyang Technological University

36

In this case, the control is returned.

## EXAMPLE 2: PRINT SOMETHING

```
void printSomething( int n )
{
    if (n <= 0)
        return;
    else {
        printf("*");
        printSomething(n-1);
        printf("!");
    }
}
```

recursive condition

The second part is the code that handles the case when the **recursive condition** occurs.

Content Copyright Nanyang Technological University

37

The second part is the code that handles the case when the recursive condition occurs.

## EXAMPLE 2: PRINT SOMETHING

```
void printSomething( int n )
{
    if (n <= 0)
        return;
    else {
        printf("*");
        printSomething(n-1);
        printf("!");
    }
}
```

recursive condition ( $n > 0$ )

The second part is the code that handles the case when the **recursive condition** occurs. This will happen when  $n$  is greater than 0.

Content Copyright Nanyang Technological University

38

This will happen when  $n$  is greater than 0.

## EXAMPLE 2: PRINT SOMETHING

```
void printSomething( int n )
{
    if (n <= 0)
        return;
    else {
        printf("*");
        printSomething(n-1);
        printf("!");
    }
}
```

recursive condition ( $n > 0$ )

printSomething() will be called again with the parameter  $n$  reduced by 1.

Content Copyright Nanyang Technological University

39

**Print Something()** will be called again with the parameter  $n$  reduced by 1.

## EXAMPLE 2: PRINT SOMETHING

```
void printSomething( int n )
{
    if (n <= 0)
        return;
    else {
        printf("*");
        printSomething(n-1);
        printf("!");
    }
}
```

recursive condition ( $n > 0$ )

After that, the character string "!" will be printed on the screen.

Content Copyright Nanyang Technological University

40

After that, the character string **exclamation mark** will be printed on the screen.

## EXAMPLE 2: PRINT SOMETHING

```
void printSomething( int n )
{
    if (n <= 0)
        return;
    else {
        printf("*");
        printSomething(n-1);
        printf("!");
    }
}
```

Note that when the function completes execution, the control will be returned to the calling function and continue execute the statement(s) after that function call.

Content Copyright Nanyang Technological University

41

Note that when the function completes execution, the control will be returned to the calling function and continue execute the statements after that function call.

## EXAMPLE 2: PRINT SOMETHING

```
void printSomething( int n )
{
    if (n <= 0)
        return;
    else {
        printf("*");
        printSomething(n-1);
        printf("!");
    }
}
```

What does the recursive function print when called with  
**printSomething(3)**?

Content Copyright Nanyang Technological University

42

The program tracing for is illustrated as follows.

## EXAMPLE 2: PRINT SOMETHING

The diagram illustrates the execution of a recursive function call. On the left, the code for `printSomething` is shown:

```
void printSomething( int n )
{
    if (n <= 0)
        return;
    else {
        printf("*");
        printSomething(n-1);
        printf("!");
    }
}
```

The recursive call `printSomething(3)` is highlighted in orange. On the right, the state of the function call stack is shown for `n=3`. The current call frame is highlighted in yellow. The output window at the bottom shows the result of the printing.

n=3

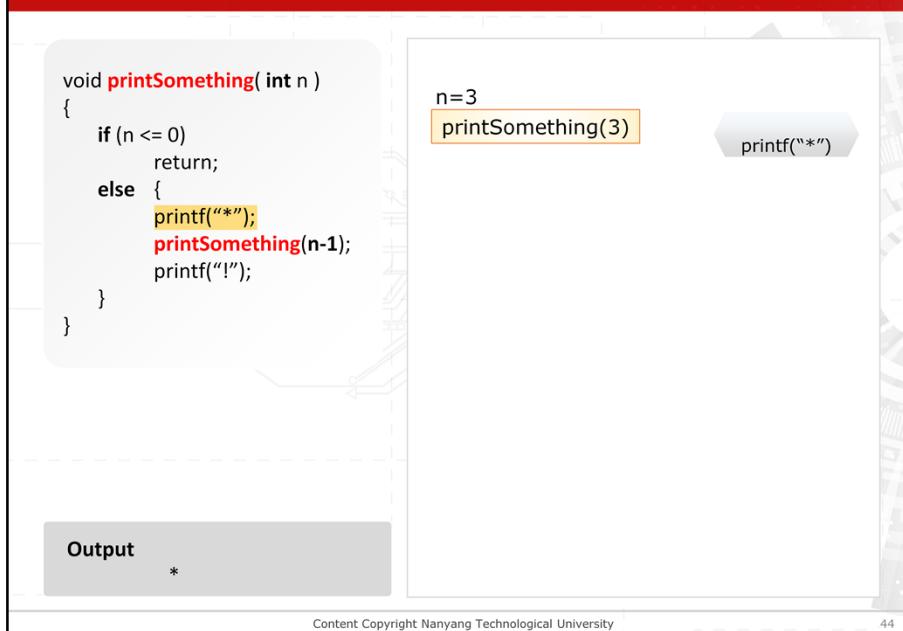
printSomething(3)

Output

Content Copyright Nanyang Technological University 43

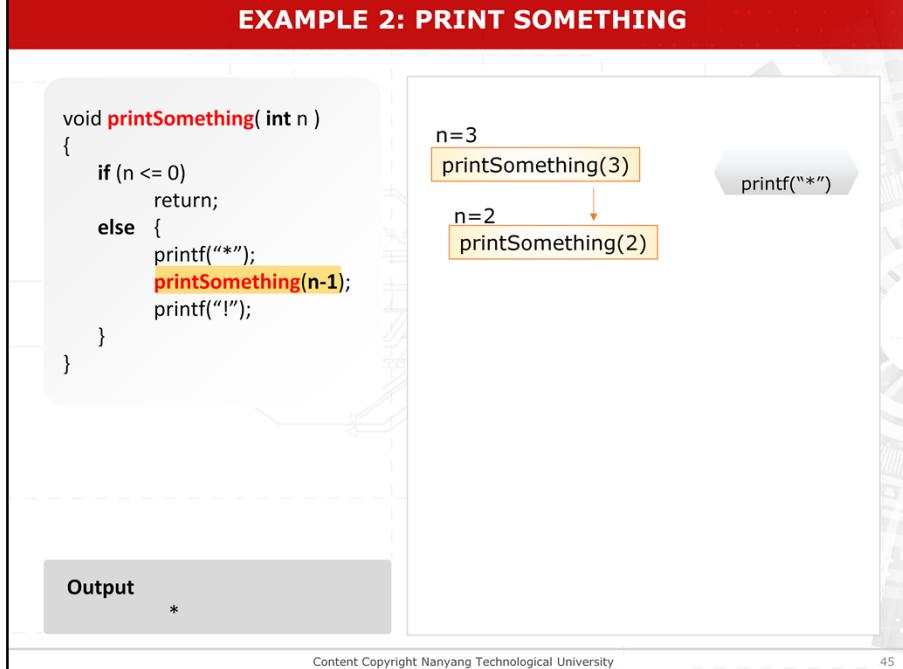
When the parameter **n** is equal to 3, the code for recursive condition is then executed.

## EXAMPLE 2: PRINT SOMETHING



The string asterisk will be printed on the screen.

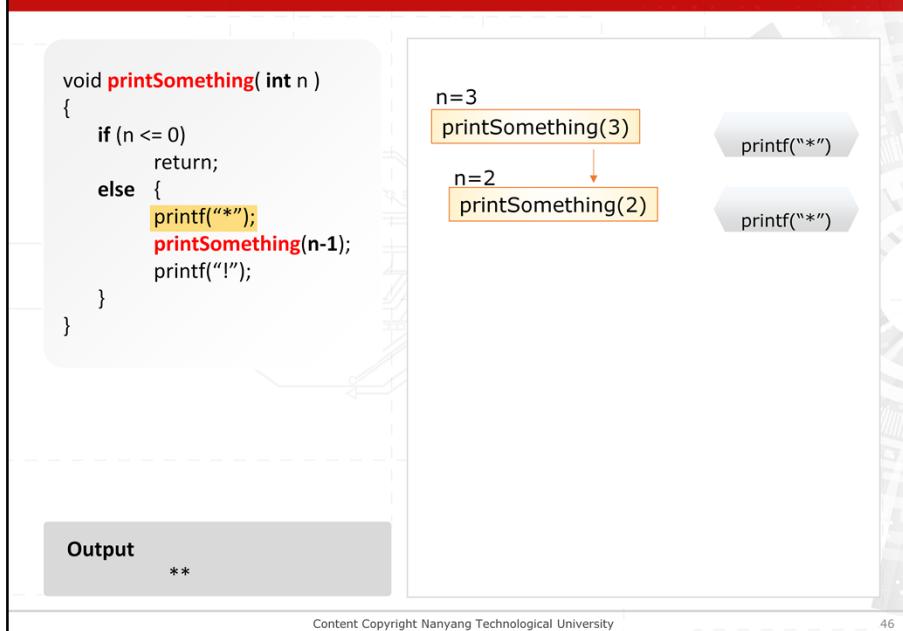
## EXAMPLE 2: PRINT SOMETHING



The function **print Something (2)** will be called with **n** is equal to 2.

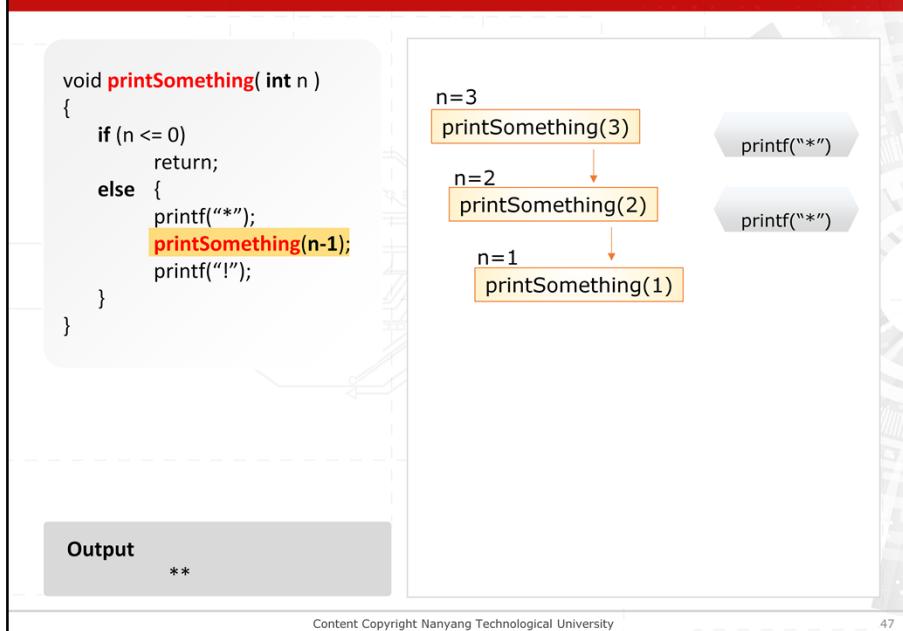
This will be repeated to print the **asterisk** for two more times until the function **print Something (0)** is called with **n** reduced to 0.

## EXAMPLE 2: PRINT SOMETHING



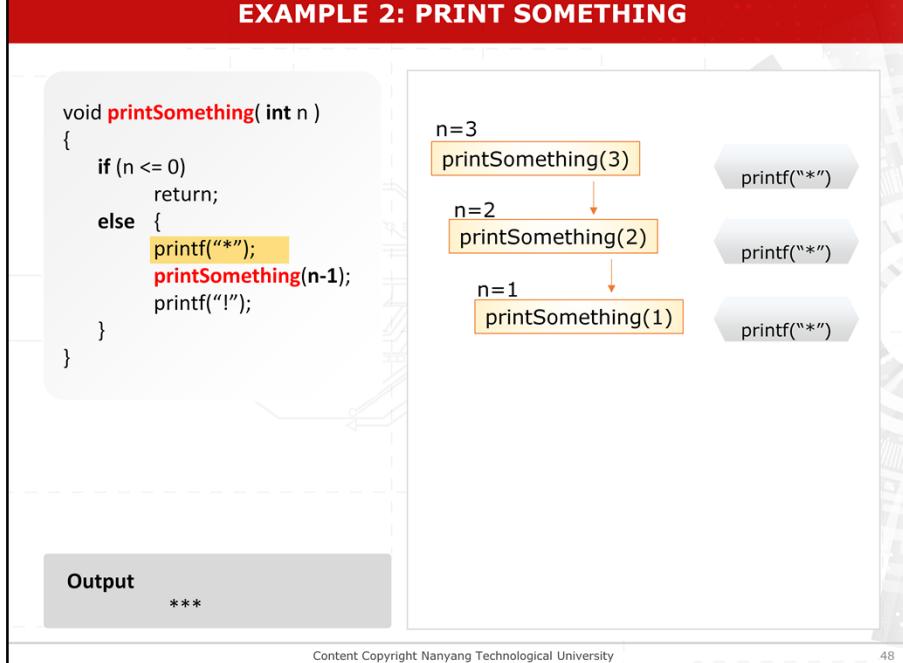
[no audio]

## EXAMPLE 2: PRINT SOMETHING



[no audio]

## EXAMPLE 2: PRINT SOMETHING

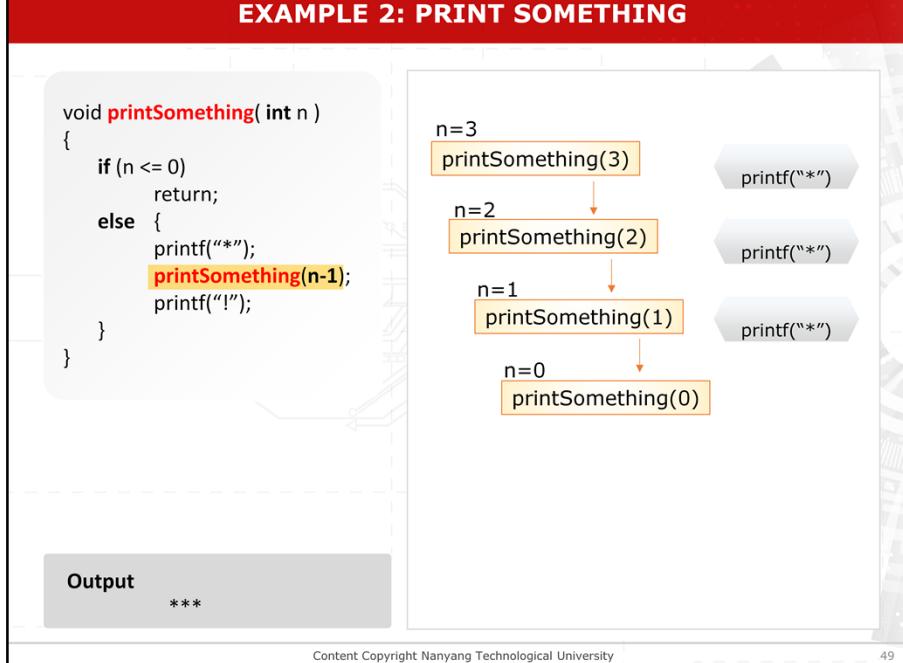


Content Copyright Nanyang Technological University

48

[no audio]

## EXAMPLE 2: PRINT SOMETHING

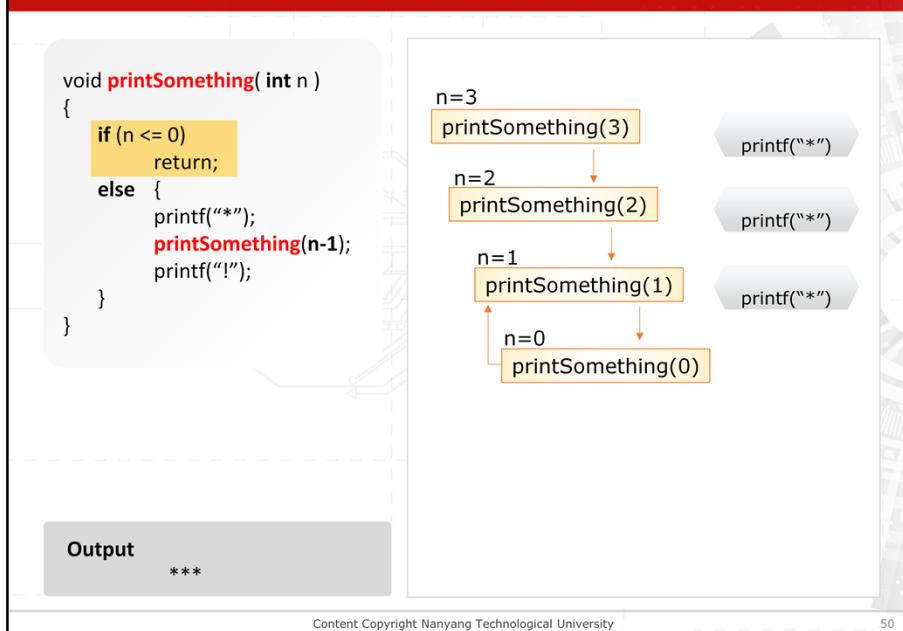


Content Copyright Nanyang Technological University

49

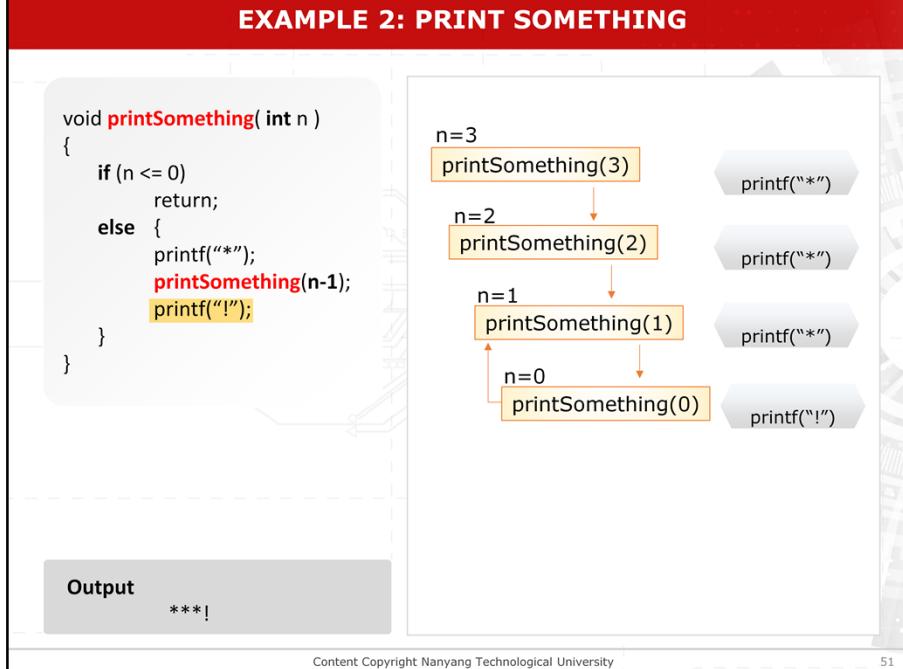
[no audio]

## EXAMPLE 2: PRINT SOMETHING



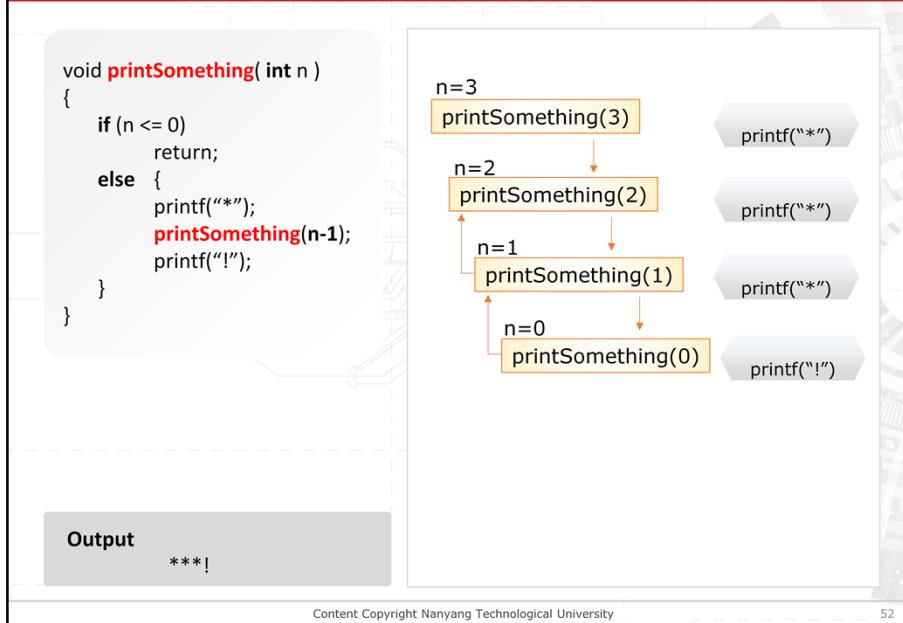
At  $n$  is equal to 0, the **return** statement will return the control to the previous function call **print Something (1)**.

## EXAMPLE 2: PRINT SOMETHING



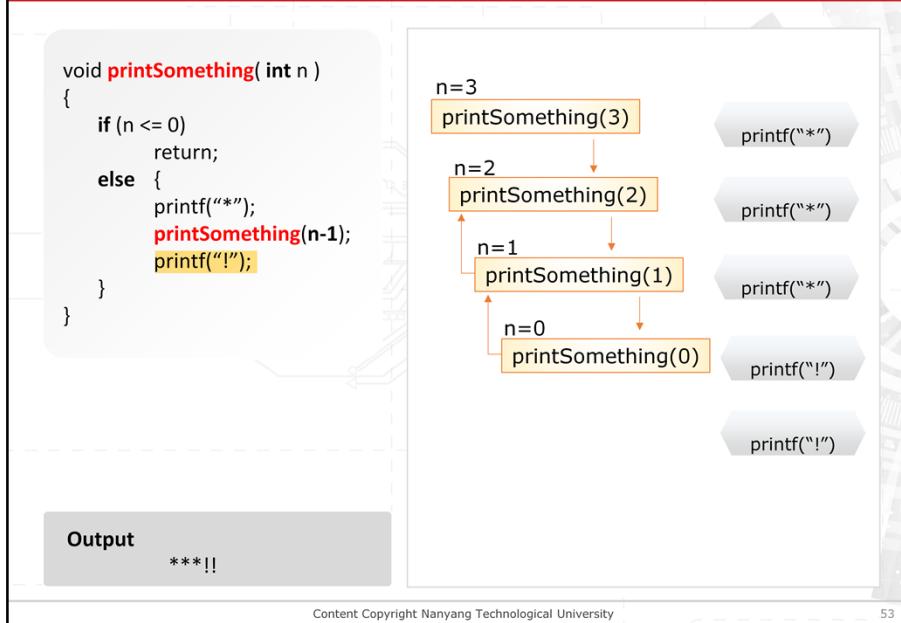
In **print Something (1)**, it will continue its execution with the next statement **printf (exclamation mark)**; and the string **exclamation mark** will be printed.

## EXAMPLE 2: PRINT SOMETHING



After that, the function has completed execution and the control is returned to the previous function call executing `print Something (2)`.

## EXAMPLE 2: PRINT SOMETHING

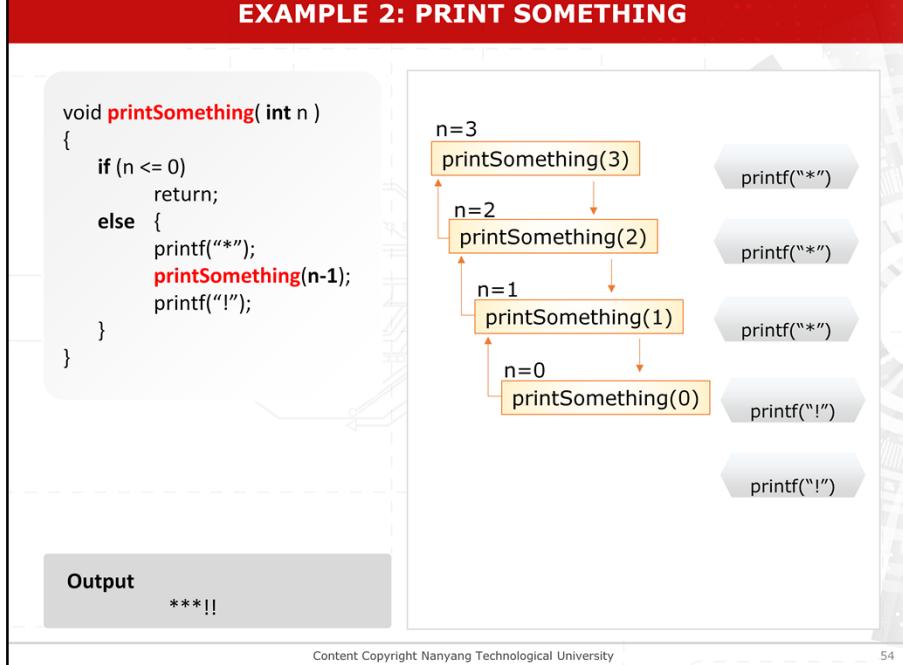


Content Copyright Nanyang Technological University

53

It will in turn complete the execution and print the string **exclamation mark**,

## EXAMPLE 2: PRINT SOMETHING

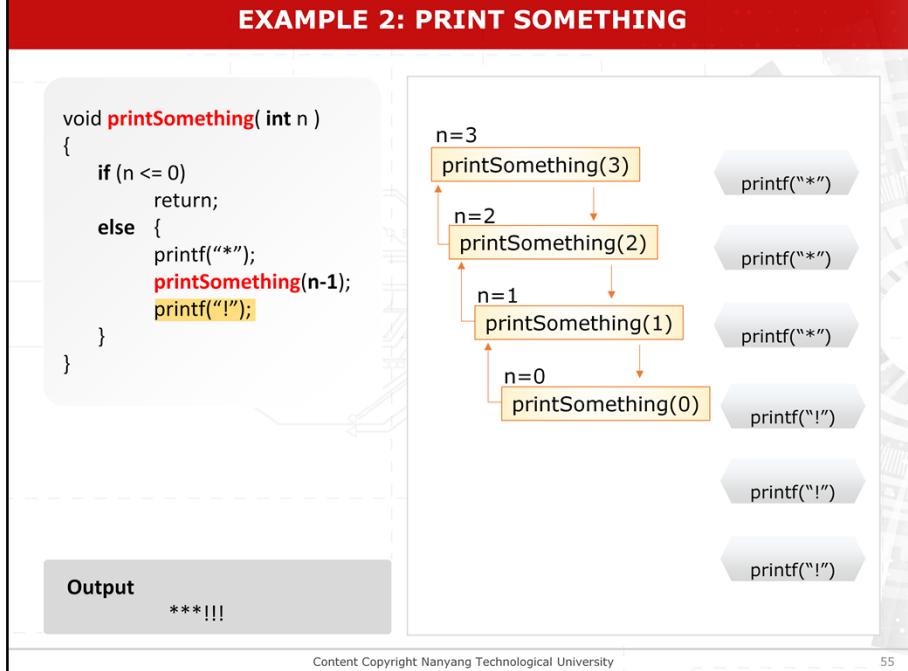


Content Copyright Nanyang Technological University

54

the control will then be returned to the previous calling function **print Something (3)**.

## EXAMPLE 2: PRINT SOMETHING



Output

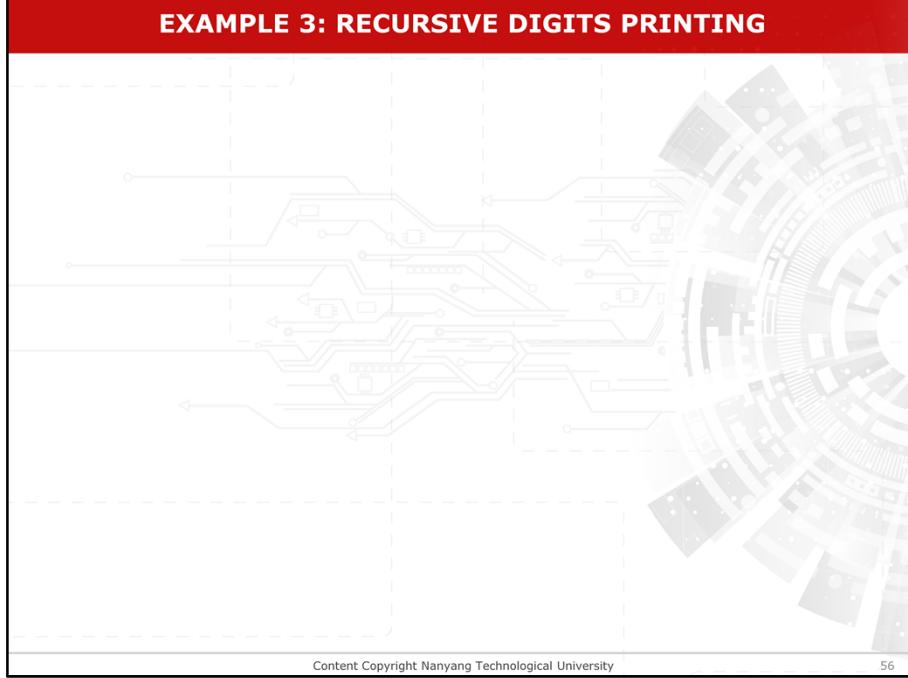
\*\*\*!!!

Content Copyright Nanyang Technological University

55

The function **print Something (3)** prints the string **exclamation mark** and returns the control to the calling **main()** function.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING



Content Copyright Nanyang Technological University

56

Example 3: Recursive Digits Printing

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

- For example: Given a number, say 2345, print each digit of the number **one digit per line**.

In this example, we write a recursive function to print the digits of an integer number one digit per line.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

- For example: Given a number, say 2345, print each digit of the number **one digit per line**.
- The recursive implementation should consist of two parts:

To write the code, we need to divide the code into two parts, one for the terminating condition, and the other for the recursive condition.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

- For example: Given a number, say 2345, print each digit of the number **one digit per line**.
- The recursive implementation should consist of two parts:
  - **Terminating Condition:** It will happen when the number is a single digit. Then, just print that digit.

Content Copyright Nanyang Technological University

59

For the **terminating condition**, it will happen when the number is a single digit. In this case, the digit is just printed on the screen.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

- For example: Given a number, say 2345, print each digit of the number **one digit per line**.
- The recursive implementation should consist of two parts:
  - **Terminating Condition:** It will happen when the number is a single digit. Then, just print that digit.
  - **Recursive Condition:** It **reduces** the problem into a **smaller** but the same problem by using integer division and modulus operators.

Content Copyright Nanyang Technological University

60

For the **recursive condition**, it will reduce the problem into a smaller but the same problem by calling the function itself again.

Here, we will need to use the division operator and modulus operator in order to obtain the quotient and remainder of a **number**.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

- Division operator (number/10)

Quotient of the number = number / 10

The division operator will be used to determine the quotient of the number because quotient = **number** divided by 10

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

- Division operator (number/10)

Quotient of the number = number / 10

- Modulus operator (number%10)

Remainder digit of the number = number % 10

the modulus operator that is digit = **number modulus 10** will give the remainder digit of the number.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

Given a number, say 2345, print each digit of the number **one digit per line**.

Content Copyright Nanyang Technological University

63

Let's look at the code that uses the division and modulus operator in recursive function such that given a number, say 2345, the code will print each digit of the number one digit per line.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
#include <stdio.h>
void printDigit(int num);
int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printDigit(num);
    return 0;
}
void printDigit(int num)
{
    if (num < 10)
        printf("%d", num);
    else {
        printDigit(num/10);
        printf( "%d\n", num%10 );
    }
}
```

Content Copyright Nanyang Technological University

64

The recursive function **print Digit()** is implemented as shown.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
#include <stdio.h>
void printDigit(int num);
int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printDigit(num);
    return 0;
}
void printDigit(int num)
{
    if (num < 10)
        printf("%d", num);
    else {
        printDigit(num/10);
        printf( "%d\n", num%10 );
    }
}
```

recursive function

Content Copyright Nanyang Technological University

65

In the recursive function **print Digit()**, it receives an integer argument **num** from the calling function.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
#include <stdio.h>
void printDigit(int num);
int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printDigit(num);
    return 0;
}
void printDigit(int num)
{
    if (num < 10)
        printf("%d", num);
    else {
        printDigit(num/10);
        printf( "%d\n", num%10 );
    }
}
```

terminating condition  
(argument num is a single digit)

Content Copyright Nanyang Technological University

66

The terminating condition is defined when the argument **num** is a single digit.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
#include <stdio.h>
void printDigit(int num);
int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printDigit(num);
    return 0;
}
void printDigit(int num)
{
    if (num < 10)
        printf("%d", num);
    else {
        printDigit(num/10);
        printf( "%d\n", num%10 );
    }
}
```

recursive condition

Content Copyright Nanyang Technological University

67

The function calls itself for the recursive condition

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
#include <stdio.h>
void printDigit(int num);
int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printDigit(num);
    return 0;
}
void printDigit(int num)
{
    if (num < 10)
        printf("%d", num);
    else {
        printDigit(num/10);
        printf( "%d\n", num%10 );
    }
}
```

recursive condition  
(argument num will be updated to  
get the quotient of num by using  
the division operator where  
num/10)

Content Copyright Nanyang Technological University

68

and each time the argument **numb** will be updated to get the quotient of **numb** by using the division operator where **numb is divided by 10**.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
#include <stdio.h>
void printDigit(int num);
int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printDigit(num);
    return 0;
}
void printDigit(int num)
{
    if (num < 10)
        printf("%d", num);
    else {
        printDigit(num/10);
        printf( "%d\n", num%10 );
    }
}
```

Content Copyright Nanyang Technological University

69

The corresponding digit of the input number is obtained through **num % modulus 10** which will be printed on the screen after the recursive call.

The recursive condition will be called repeatedly until the terminating condition is reached. In which case, the final digit of the input number will be printed on the screen.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
#include <stdio.h>
void printDigit(int num);
int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printDigit(num);
    return 0;
}
void printDigit(int num)
{
    if (num < 10)
        printf("%d", num);
    else {
        printDigit(num/10);
        printf( "%d\n", num%10 );
    }
}
```

#### Output

Enter a number: 2345

Content Copyright Nanyang Technological University

70

The program tracing for **print Digit (2345)** will be illustrated as follow.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
#include <stdio.h>
void printDigit(int num);
int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printDigit(num);
    return 0;
}
void printDigit(int num)
{
    if (num < 10)
        printf("%d", num);
    else {
        printDigit(num/10);
        printf( "%d\n", num%10 );
    }
}
```

#### Output

Enter a number: 2345

Content Copyright Nanyang Technological University

71

We will look into detail on the recursive function print Digit() for numb is equal to 2345

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
void printDigit(int num) {  
    if (num < 10)  
        printf("%d", num);  
    else {  
        printDigit(num / 10);  
        printf( "%d\n", num%10 );  
    }  
}
```

#### Output

Enter a number: 2345

Content Copyright Nanyang Technological University

72

[no audio]

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
void printDigit(int num) {  
    if (num < 10)  
        printf("%d", num);  
    else {  
        printDigit(num / 10);  
        printf( "%d\n", num%10 );  
    }  
}
```

num=2345

printDigit(2345)

#### Output

Enter a number: 2345

Content Copyright Nanyang Technological University

73

When the parameter **numb** is equal to 2345, the code for recursive condition is then executed.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
void printDigit(int num) {  
    if (num < 10)  
        printf("%d", num);  
    else {  
        printDigit(num / 10);  
        printf( "%d\n", num%10 );  
    }  
}
```

#### Output

Enter a number: 2345

num=2345

printDigit(2345)

num=234

quotient num = 2345/10 = 234

Content Copyright Nanyang Technological University

74

The function **print Digit (numb divided by 10)** will be called with the quotient **numb is equal to 2345** divided by 10 is equal to **234**.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
void printDigit(int num) {  
    if (num < 10)  
        printf("%d", num);  
    else {  
        printDigit(num / 10);  
        printf( "%d\n", num%10 );  
    }  
}
```

#### Output

Enter a number: 2345

num=2345  
printDigit(2345)

num=234  
printDigit(234)

num=23

quotient num = 234/10 = 23

Content Copyright Nanyang Technological University

75

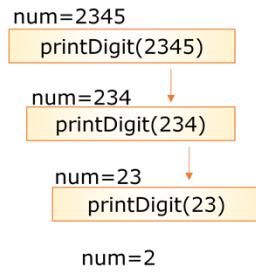
Again the function **print Digit (num divided by 10)** will be called with the quotient **numb** is equal to 234 divided by 10 is equal to **23**.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
void printDigit(int num) {  
    if (num < 10)  
        printf("%d", num);  
    else {  
        printDigit(num / 10);  
        printf( "%d\\n", num%10 );  
    }  
}
```

#### Output

Enter a number: 2345



quotient num = 23/10 = 2

Content Copyright Nanyang Technological University

76

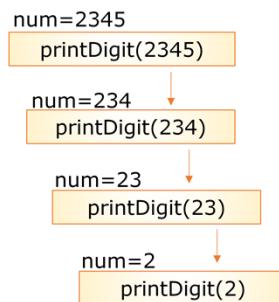
The function **print Digit (numb divided by 10)** will be called with the quotient **numb** is equal to 23 divided by 10 is equal to **2**.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
void printDigit(int num) {  
    if (num < 10)  
        printf("%d", num);  
    else {  
        printDigit(num / 10);  
        printf( "%d\\n", num%10 );  
    }  
}
```

#### Output

Enter a number: 2345



Content Copyright Nanyang Technological University

77

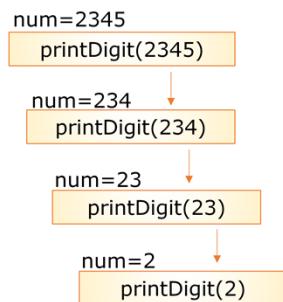
Similarly, the function **print Digit (2)** is called again.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
void printDigit(int num) {  
    if (num < 10)  
        printf("%d", num);  
    else {  
        printDigit(num / 10);  
        printf(" %d\\n", num%10 );  
    }  
}
```

#### Output

Enter a number: 2345



reaches the terminating condition with only a single digit

Content Copyright Nanyang Technological University

78

When this function is called, it reaches the terminating condition with only a single digit.

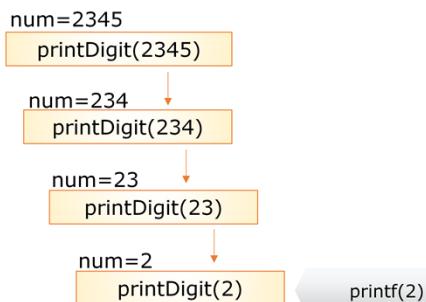
### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
void printDigit(int num) {  
    if (num < 10)  
        printf("%d", num);  
    else {  
        printDigit(num / 10);  
        printf(" %d\\n", num%10 );  
    }  
}
```

#### Output

Enter a number: 2345

2



Content Copyright Nanyang Technological University

79

The value 2 is printed to the screen.

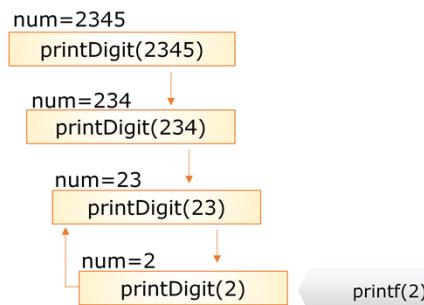
### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
void printDigit(int num) {  
    if (num < 10)  
        printf("%d", num);  
    else {  
        printDigit(num / 10);  
        printf(" %d\\n", num%10);  
    }  
}
```

#### Output

Enter a number: 2345

2



Content Copyright Nanyang Technological University

80

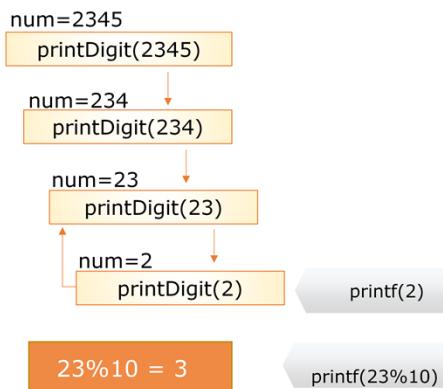
The control is then returned to the previous calling function **print Digit (23)**.

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
void printDigit(int num) {  
    if (num < 10)  
        printf("%d", num);  
    else {  
        printDigit(num / 10);  
        printf(" %d\\n", num%10);  
    }  
}
```

#### Output

```
Enter a number: 2345  
2  
3
```



Content Copyright Nanyang Technological University

81

The function continues execution and prints the digit 3 to the screen because  $23 \bmod 10$  is equal to 3.

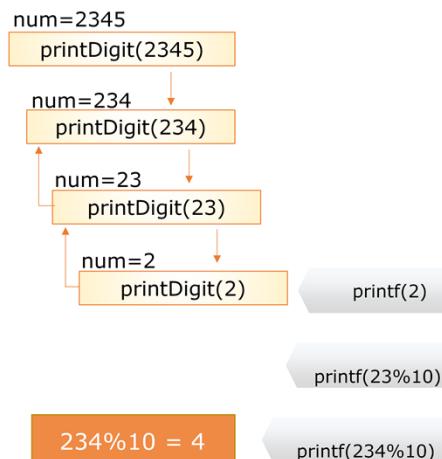
### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
void printDigit(int num) {  
    if (num < 10)  
        printf("%d", num);  
    else {  
        printDigit(num / 10);  
        printf(" %d\\n", num%10);  
    }  
}
```

#### Output

Enter a number: 2345

2  
3  
4



Content Copyright Nanyang Technological University

82

Similarly, the digits 4 and 5 will then be printed when the control is returned to the previous calling functions.

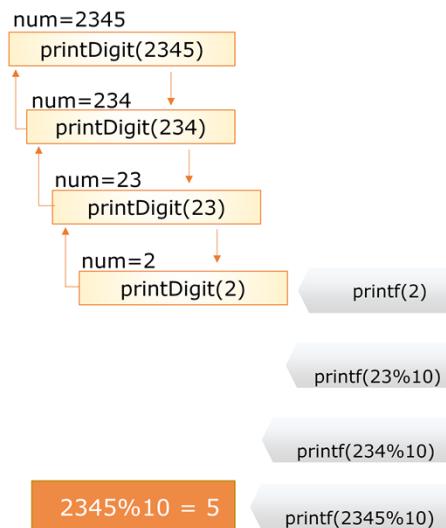
### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
void printDigit(int num) {  
    if (num < 10)  
        printf("%d", num);  
    else {  
        printDigit(num / 10);  
        printf(" %d\\n", num%10);  
    }  
}
```

#### Output

Enter a number: 2345

2  
3  
4  
5



Content Copyright Nanyang Technological University

83

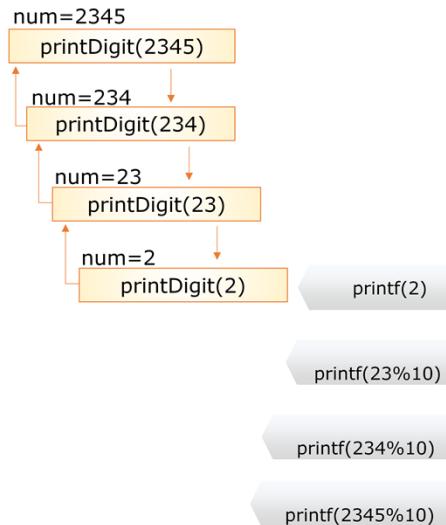
[no audio]

### EXAMPLE 3: RECURSIVE DIGITS PRINTING

```
void printDigit(int num) {  
    if (num < 10)  
        printf("%d", num);  
    else {  
        printDigit(num / 10);  
        printf(" %d\\n", num%10 );  
    }  
}
```

#### Output

```
Enter a number: 2345  
2  
3  
4  
5
```



Content Copyright Nanyang Technological University

84

In this program, it is important to observe the execution order of the statements when the function has completed execution.

The control will be returned to the previous calling function and continues execution. It is also important to observe that the **printf()** statement is placed after the recursive function call in this program.

And also observe that each call to a function has its own set of values for the actual arguments and local variables.

## SUMMARY

After this lesson, you should be able to:

- Explain recursion and how it works  
with some examples

Content Copyright Nanyang Technological University

85

In summary, after viewing this video lesson, you should be able to explain recursion and how it works with some examples.

We will look into more examples of recursion in the next video lecture.