UNIVERSITI TEKNOLOGI MARA

CSC508 Data Structures

Topic 7 : Recursion

Compiled & edited by: Zahid Zainal

UNIVERSITI TEKNOLOGI MARA

Recap

- Queue Definition
- Queue Operations
- Queue Implementation
- Circular Queue

Topic Structure

- Recursion Definition
- Recursive Method
- Types of Recursion
- Recursion Application
- Recursion vs Loop

Learning Outcomes

- At the end of this lesson, students should be able to:
  - Describe the recursion concept
  - Explain recursion implementation
  - Discuss recursion application

Recursion Definition

- Recursion is a Process of solving a problem by reducing it to smaller versions of itself.
- Recursive definition
  - Definition in which a problem is expressed in terms of a smaller version of itself.
  - Has one or more base cases.
- Recursive algorithm
  - Algorithm that finds the solution to a given problem by reducing the problem to smaller versions of itself.
  - Has one or more base cases.
  - Implemented using recursive methods.

Recursive Method

- Method that calls itself
- A recursive method contains:
  - Base case
    - Case in recursive definition in which the solution is obtained directly.
    - Stops the recursion.
  - General case
    - Breaks problem into smaller versions of itself.
    - Case in recursive definition in which a smaller version of itself is called.
    - Must eventually be reduced to a base case.

Recursive Method (cont.)

- Has infinite copy of itself
- Every recursive call has
  - its own code
  - own set of parameters
  - own set of local variables

Tracing A Recursive Method

- After completing recursive call:
  - Control goes back to calling environment.
  - Recursive call must execute completely before control goes back to previous call.
  - Execution in previous call begins from point immediately following recursive call.

Type of Recursion

- **Direct recursion**: a method that calls itself
- **Indirect recursion**: a method that calls another method and eventually results in the original method call.
- **Tail recursion**: recursive method in which the last statement executed is the recursive call.
- **Infinite recursion**: the case where every recursive call results in another recursive call. No base case

```
void fun1(int a) {
        if (a > 0) {
                printf("%d\n", a);
                fun2 (a - 1);
        }
}

void fun2(int b) {
        if (b > 0) {
                printf("%d\n", b);
                fun1 (b - 3);
        }
}

int main() {
        int v = 12;
        fun1(v);
}
```

Designing Recursive Method

- Understand problem requirements
- Determine limiting conditions
- Identify base cases
- Provide direct solution to each base case
- Identify general case(s)
- Provide solutions to general cases in terms of smaller versions of itself

Recursion Application

- Factorial Number
- Fibonacci number
- Linked list reverse print
- Binary tree traversal
- Merge sort

Factorial Number

- The factorial of a positive integer is the number multiplied by every positive integer less than itself
- Example :

  3! = 3 * 2! = 6

  2! = 2 * 1! = 2

  1! = 1 * 0! = 1

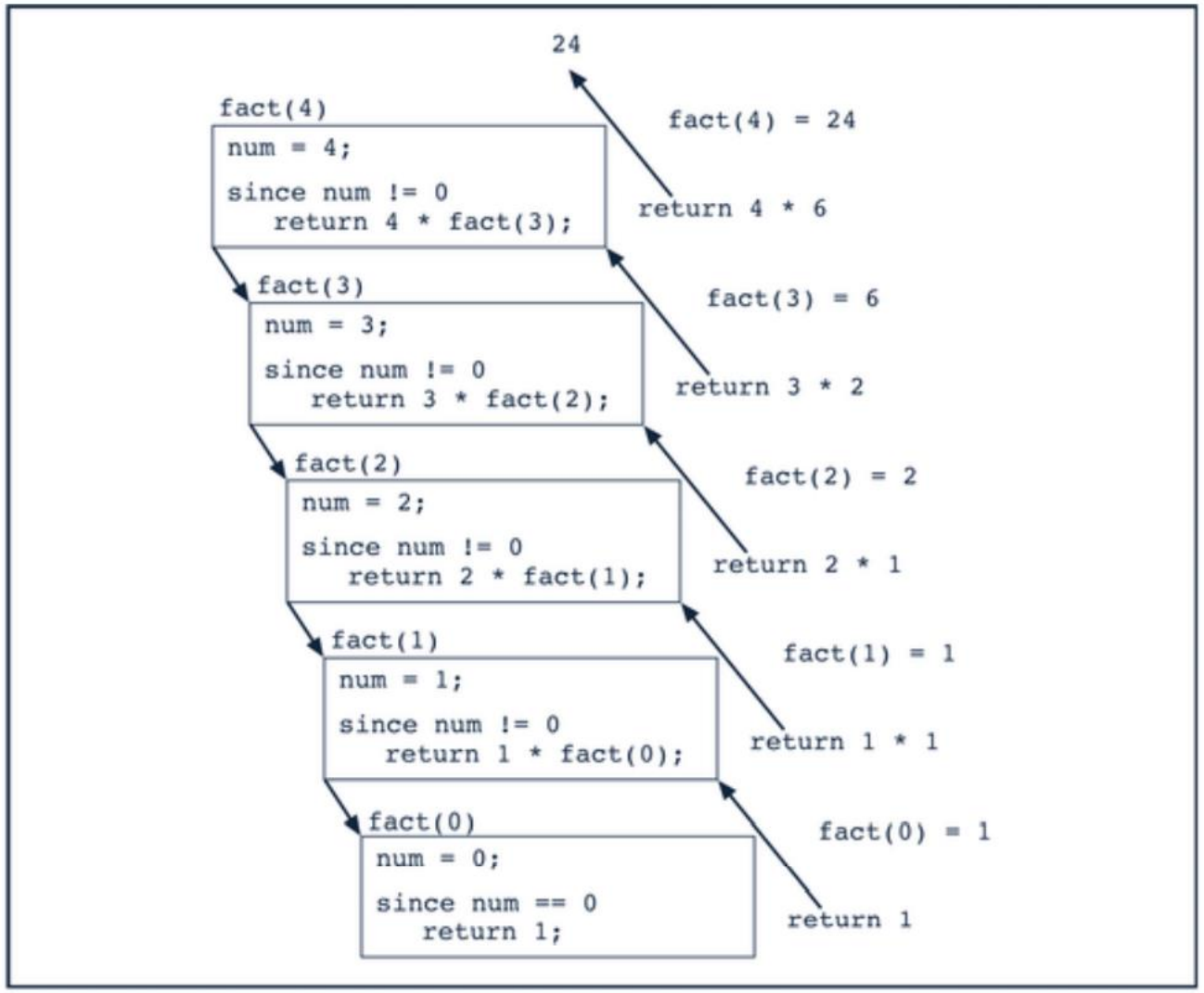
  0! = 1

```
public class Recursion {
    public static int fact(int x) {
        if (x == 0)
            return 1;
        else {
            System.out.println("x = " + x);
            return x * fact(x-1);
        }
    }
    public static void main(String[] args) {
        int num = fact(4);
        System.out.println("Factorial : " + num);
    }
}
```

Base case

```
x = 4
x = 3
x = 2
x = 1
Factorial : 24
```

Tracing Factorial



Fibonacci Number

- Fibonacci sequence:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ……

- The third Fibonacci number is the sum of the first two Fibonacci numbers.
- The fourth Fibonacci number in a sequence is the sum of the second and third Fibonacci numbers. And so on…

Finding Fibonacci

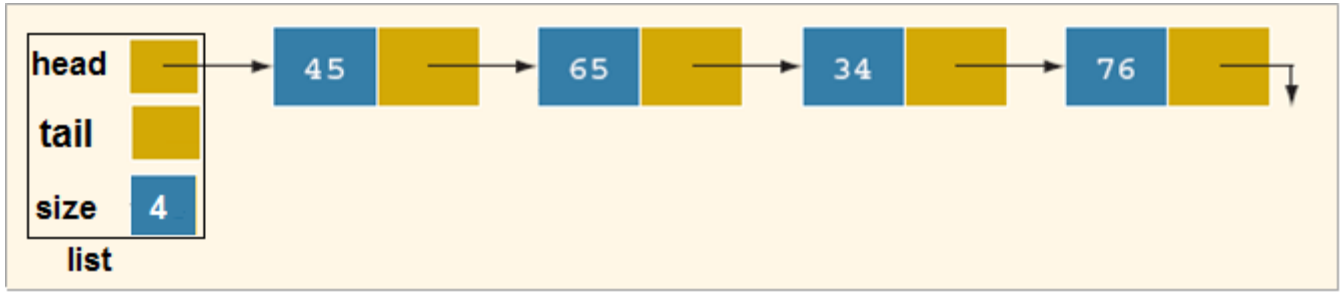
```
public class Recursion {
    public static int fib(int n){
        if(n == 0)
            return 0;
        else if(n == 1)
            return 1;
        else
            return fib(n - 1) + fib(n - 2);
    }

    public static void main(String[] args) {
        int num = fib(6);
        System.out.println("Fib : " + num);
    }
}
```

Base case

Base case

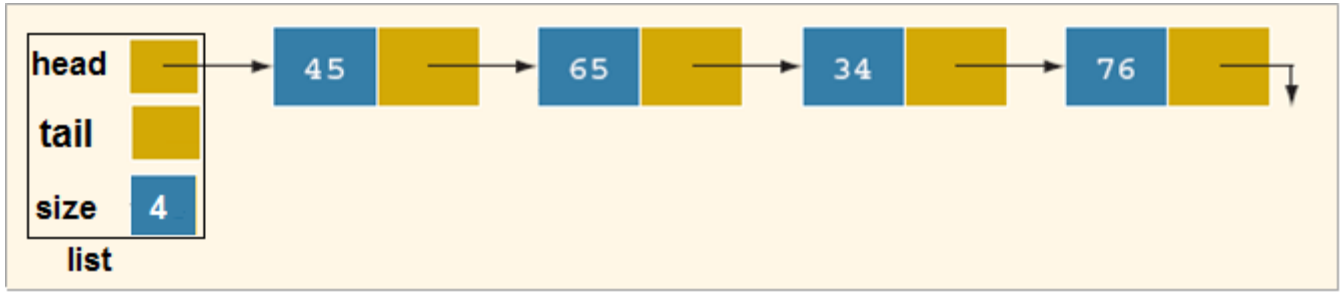
General case

Linked List Reverse Print




```
public void printReverse(){
    printRecursive(head);
}

public void printRecursive(Node temp){
    if(tempt != null) {
        printRecursive(temp.next);
        System.out.println(temp.data);
    }
}
```

Tracing Reverse Print



printRecursive of node 45, calls…

printRecursive of node 65, calls…

printRecursive of node 34, calls…

printRecursive of node 76, calls…

printRecursive of null node, no further calls

recursion ends, execution returns to caller

prints 76, return to caller

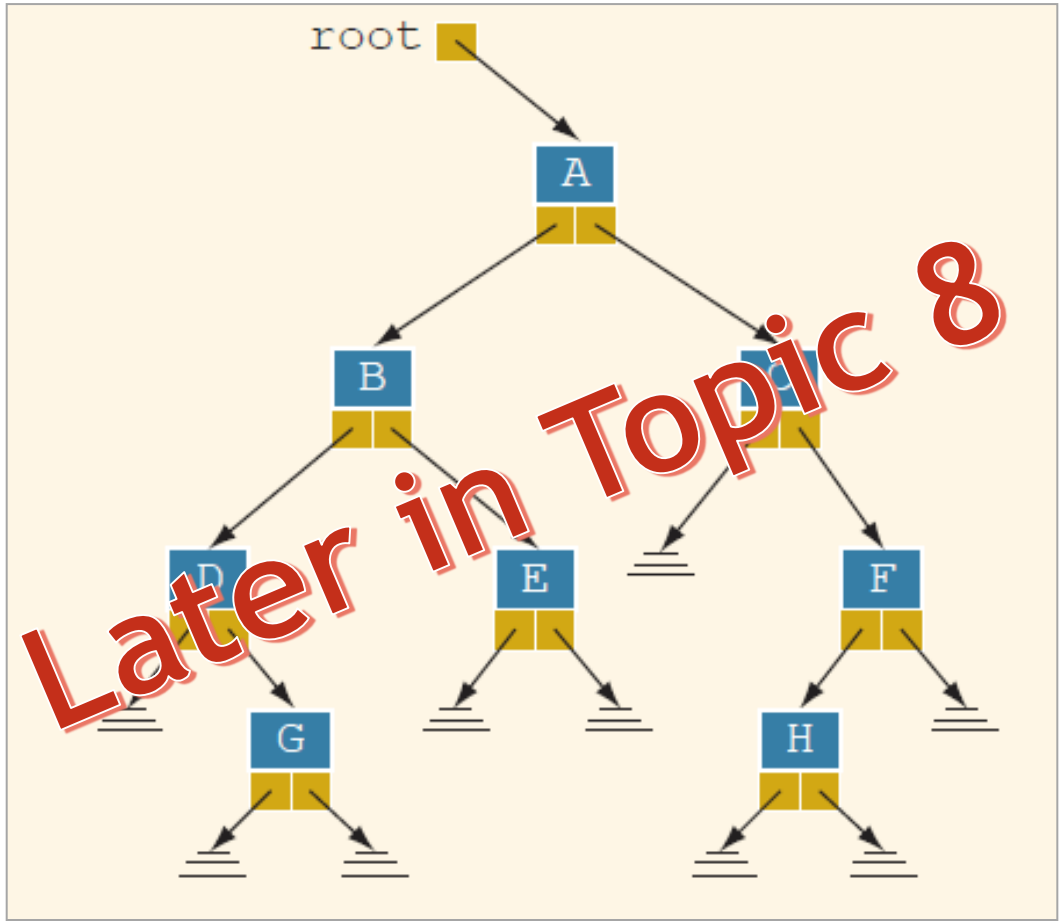
prints 34, return to caller

prints 65, return to caller

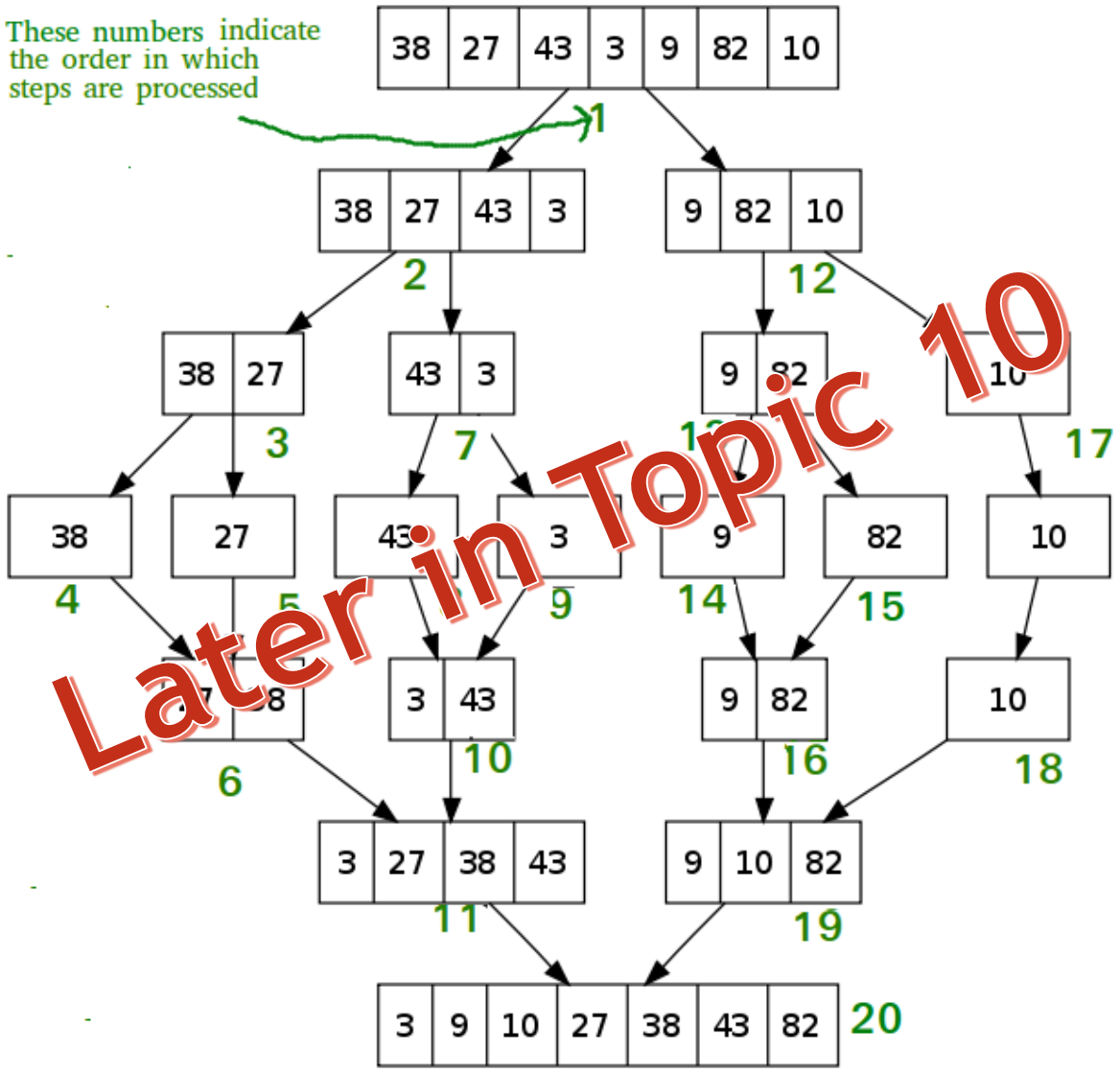
prints 45, done. return to caller

```
public void printReverse(){
    printRecursive(head);
}

public void printRecursive(Node temp){
    if(tempt != null) {
        printRecursive(temp.next);
        System.out.println(temp.data);
    }
}
```

Binary Tree Traversal



Merge Sort


These numbers indicate the order in which steps are processed
38 27 43 3 9 82 10
1
38 27 43 3
9 82 10
2
12
38 27
43 3
9 82
10
3
7
17
38
27
43
3
9
82
10
4
9
14
15
3 43
9 82
10
10
16
18
3 27 38 43
9 10 82
11
19
3 9 10 27 38 43 82
20
Later in Topic 10

Recursion vs Loop

RECURSION	LOOP
A method of calling a function within the same function	A control structure that allows executing a block of code repeatedly within the program
Execution is slower	Execution is faster
Stack is used to store the local variables when the function is called	Stack is not used
If there is no termination condition, it can be an infinite recursion	If the condition never becomes false, it will be an infinite loop
More readable	Less readable
Space complexity is higher	Space complexity is lower
	Visit www.PEDIAA.com

https://pediaa.com/what-is-the-difference-between-recursion-and-loop/

Summary

- Recursion is a Process of solving a problem by reducing it to smaller versions of itself.
- Recursive method is a method that calls itself
- Recursion types : Direct recursion, Indirect recursion, Tail recursion, Infinite recursion
- Recursion has higher space and time complexity

Next Topic...

- Tree

References

- Carrano, F. & Savitch, W. 2005. *Data Structures and Abstractions with Java, 2nd ed. Prentice-Hall.*
- Malik D.S, & Nair P.S., Data Structures Using Java, Thomson Course Technology, 2003.
- Rada Mihalcea, CSCE 3110 Data Structures and Algorithm Analysis notes, U of North Texas.