

CSE1301  
Computer Programming  
Lecture 27  
*Recursion (Part 1)*

1

Topics

- Recursion
- Unary Recursion
- *N*-ary Recursion

2

Recursion

- Queue processing
- Sorting
- Searching

3

What is Recursion?

- A procedure defined in terms of (simpler versions of) itself
- Components:
  - Base case
  - Recursive definition
  - Convergence to base case

4

Example: Queue processing

```
procedure ProcessQueue ( queue )
{
  if ( queue not empty ) then
  {
    process first item in queue
    remove first item from queue
    ProcessQueue ( rest of queue )
  }
}
```

5

Example: Queue processing

```
procedure ProcessQueue ( queue )
{
  if ( queue not empty ) then
  {
    process first item in queue
    remove first item from queue
    ProcessQueue ( rest of queue )
  }
}
```

**Base case: queue is empty**

### Example: Queue processing

```

procedure ProcessQueue ( queue )
{
  if ( queue not empty ) then
  {
    process first item in queue
    remove first item from queue
    ProcessQueue ( rest of queue )
  }
}
    
```

**Recursion:** call to ProcessQueue

### Example: Queue processing

```

procedure ProcessQueue ( queue )
{
  if ( queue not empty ) then
  {
    process first item in queue
    remove first item from queue
    ProcessQueue ( rest of queue )
  }
}
    
```

**Convergence:** fewer items in queue

### Unary Recursion

- Functions calls itself once (at most)
- Usual format:
 

```

function RecursiveFunction ( <parameter(s)> )
{
  if ( <base case> ) then
    return <base value>
  else
    return RecursiveFunction ( <expression> )
}
      
```
- Winding and unwinding the “stack frames”

9

### Example: Factorial

Given  $n \geq 0$ :

$$n! = n \times (n - 1) \times (n - 2) \times \dots \times 2 \times 1$$

$$0! = 1$$

Example:

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

10

### Example: Factorial

- *Problem:* Write a recursive function **Factorial(n)** which computes the value of  $n!$
- *Base Case:*  
 If  $n = 0$  or  $n = 1$ :  
 Factorial( $n$ ) = 1

11

### Example: Factorial

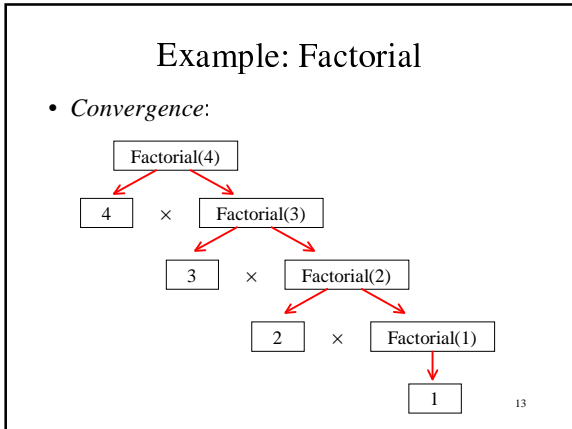
- *Recursion:*

$$n! = n \times \underbrace{(n - 1) \times (n - 2) \times \dots \times 2 \times 1}_{(n - 1)!}$$

If  $n > 1$ :

$$\text{Factorial}(n) = n \times \text{Factorial}(n - 1)$$

12



### Example: Factorial

The Factorial function can be defined recursively as follows:

Factorial(0) = 1

Factorial(1) = 1

Factorial(n) = n × Factorial(n - 1)

14

### Example: Factorial

```

function Factorial ( n )
{
  if ( n is less than or equal to 1 ) then
    return 1
  else
    return n × Factorial ( n - 1 )
}
  
```

15

### Example: Factorial

```

function Factorial ( n )
{
  if ( n is less than or equal to 1 ) then
    return 1
  else
    return n × Factorial ( n - 1 )
}
  
```

16

### Example: Factorial

```

function Factorial ( n )
{
  if ( n is less than or equal to 1 ) then
    return 1
  else
    return n × Factorial ( n - 1 )
}
  
```

17

### Example: Factorial

```

function Factorial ( n )
{
  if ( n is less than or equal to 1 ) then
    return 1
  else
    return n × Factorial ( n - 1 )
}
  
```

18

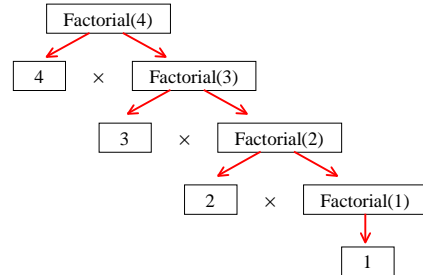
### Example: Factorial

```
function Factorial ( n )
{
  if ( n is less than or equal to 1 ) then
    return 1
  else
    return n × Factorial ( n - 1 )
}
```

**Convergence**

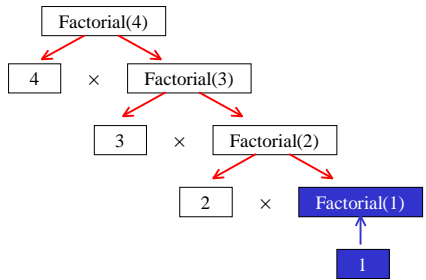
19

### Example: Factorial



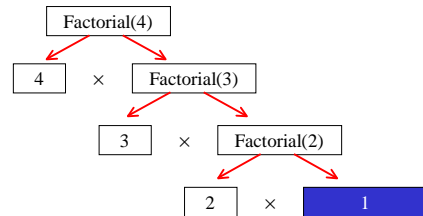
20

### Example: Factorial



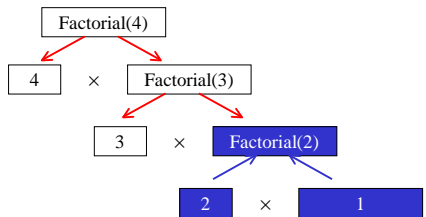
21

### Example: Factorial



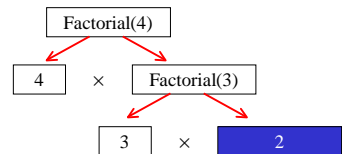
22

### Example: Factorial

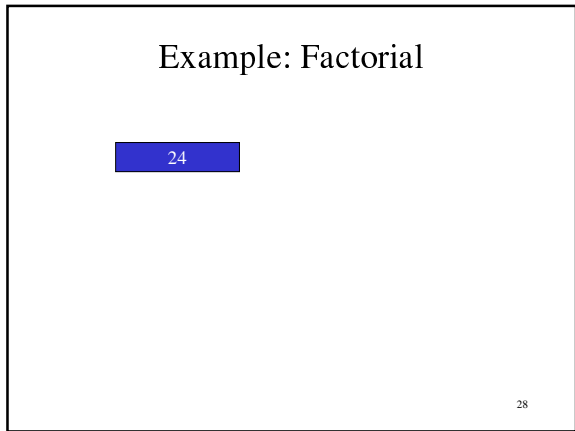
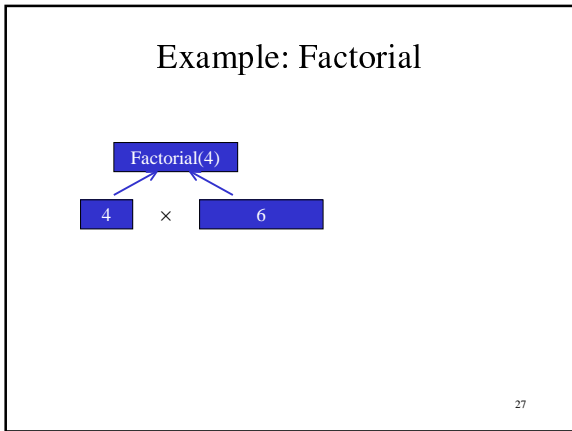
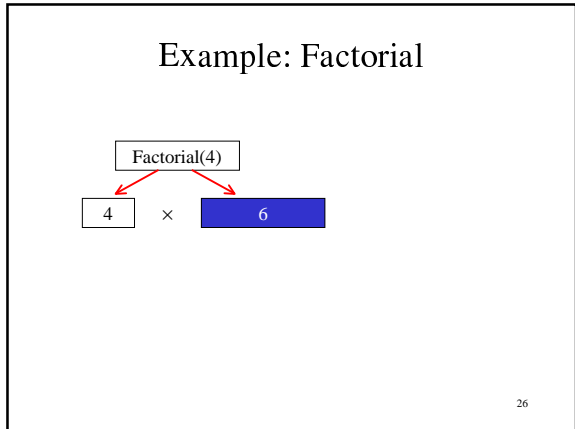
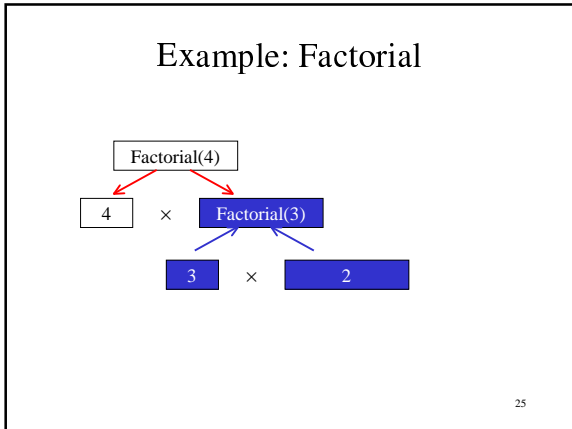


23

### Example: Factorial



24



*Example: factorial.c*

<pre> Computes the factorial of a number function Factorial ( n ) {   if ( n is less than or equal to 1 )   then     return 1   else     return n x Factorial ( n - 1 ) }         </pre>	<pre> /* Compute the factorial of n */ int factorial ( int n ) {   if ( n &lt;= 1 )   {     return 1;   }   else   {     return n * factorial (n-1);   } }         </pre>
--	---

29

*Example: "Frames" during calculation of factorial(4)*

```
printf("%d", factorial(4));
```

30

Example: "Frames" during calculation of **factorial(4)**

```
printf("%d", factorial(4));

int factorial ( int n )
{
  if ( 4 <= 1 )
  {
    return 1;
  }
  else
  {
    return 4 * factorial( 4 - 1 );
  }
}
```

31

Example: "Frames" during calculation of **factorial(4)**

```
printf("%d", factorial(4));

int factorial ( int n )
{
  if ( 4 <= 1 )
  {
    return 1;
  }
  else
  {
    return 4 * factorial( 3 );
  }
}
```

32

Example: "Frames" during calculation of **factorial(4)**

```
printf("%d", factorial(4));

int factorial ( int n )
{
  if ( 3 <= 1 )
  {
    return 1;
  }
  else
  {
    return 3 * factorial( 3 - 1 );
  }
}
```

33

Example: "Frames" during calculation of **factorial(4)**

```
printf("%d", factorial(4));

int factorial ( int n )
{
  if ( 3 <= 1 )
  {
    return 1;
  }
  else
  {
    return 3 * factorial( 2 );
  }
}
```

34

Example: "Frames" during calculation of **factorial(4)**

```
printf("%d", factorial(4));

int factorial ( int n )
{
  if ( 2 <= 1 )
  {
    return 1;
  }
  else
  {
    return 2 * factorial( 2 - 1 );
  }
}
```

35

Example: "Frames" during calculation of **factorial(4)**

```
printf("%d", factorial(4));

int factorial ( int n )
{
  if ( 2 <= 1 )
  {
    return 1;
  }
  else
  {
    return 2 * factorial( 1 );
  }
}
```

36

*Example:* "Frames" during calculation of **factorial(4)**

```

printf("%d", factorial(4));
int factorial ( int n )
{
  if ( 1 <= 1 )
  {
    return 1;
  }
  else
  {
    return n * factorial ( n - 1 );
  }
}
    
```

n: 1

37

*Example:* "Frames" during calculation of **factorial(4)**

```

printf("%d", factorial(4));
int factorial ( int n )
{
  if ( 1 <= 1 )
  {
    return 1;
  }
  else
  {
    return n * factorial ( n - 1 );
  }
}
    
```

n: 1

**Base case:**  
factorial(1) is 1

38

*Example:* "Frames" during calculation of **factorial(4)**

```

printf("%d", factorial(4));
int factorial ( int n )
{
  if ( 2 <= 1 )
  {
    return 1;
  }
  else
  {
    return 2 * factorial ( 1 );
  }
}
    
```

n: 2

1

39

*Example:* "Frames" during calculation of **factorial(4)**

```

printf("%d", factorial(4));
int factorial ( int n )
{
  if ( 2 <= 1 )
  {
    return 1;
  }
  else
  {
    return 2 * factorial ( 2 - 1 );
  }
}
    
```

n: 2

1

40

*Example:* "Frames" during calculation of **factorial(4)**

```

printf("%d", factorial(4));
int factorial ( int n )
{
  if ( 2 <= 1 )
  {
    return 1;
  }
  else
  {
    return 2;
  }
}
    
```

n: 2

factorial(2) is 2

2

41

*Example:* "Frames" during calculation of **factorial(4)**

```

printf("%d", factorial(4));
int factorial ( int n )
{
  if ( 3 <= 1 )
  {
    return 1;
  }
  else
  {
    return 3 * factorial ( 2 );
  }
}
    
```

n: 3

2

42

Example: "Frames" during calculation of **factorial(4)**

```
printf("%d", factorial(4));
```

```
int factorial ( int n ) {
  if ( 3 <= 1 )
    return 1;
  else
    return 3 * factorial( 3 - 1 );
}
```

n: 3

2

43

Example: "Frames" during calculation of **factorial(4)**

```
printf("%d", factorial(4));
```

```
int factorial ( int n ) {
  if ( 3 <= 1 )
    return 1;
  else
    return 6 ;
}
```

n: 3

factorial(3) is 6

6

44

Example: "Frames" during calculation of **factorial(4)**

```
printf("%d", factorial(4));
```

```
int factorial ( int n ) {
  if ( 4 <= 1 )
    return 1;
  else
    return 4 * factorial( 3 );
}
```

n: 4

6

45

Example: "Frames" during calculation of **factorial(4)**

```
printf("%d", factorial(4));
```

```
int factorial ( int n ) {
  if ( 4 <= 1 )
    return 1;
  else
    return 4 * factorial( 4 - 1 );
}
```

n: 4

6

46

Example: "Frames" during calculation of **factorial(4)**

```
printf("%d", factorial(4));
```

```
int factorial ( int n ) {
  if ( 4 <= 1 )
    return 1;
  else
    return 24 ;
}
```

n: 4

factorial(4) is 24

24

47

Example: "Frames" during calculation of **factorial(4)**

```
printf("%d", factorial(4));
```

24

Output: 24

48

Example: testprog.c

```

#include <stdio.h>
#include "factor1.c"

/* Main program for testing factorial() function */

int main(void)
{
    int n;

    printf("Please enter n: ");
    scanf("%d", &n);

    printf("%d! is %d\n", n, factorial(n));

    return 0;
}
    
```

### N-ary Recursion

- Sometimes a function can only be defined in terms of two or more calls to itself.
- Efficiency is often a problem.

### Example: Fibonacci

- A series of numbers which
  - begins with 0 and 1
  - every subsequent number is the sum of the previous two numbers
- 0, 1, 1, 2, 3, 5, 8, 13, 21,...
- Write a recursive function which computes the *n*-th number in the series (*n* = 0, 1, 2,...)

### Example: Fibonacci

The Fibonacci series can be defined recursively as follows:

Fibonacci(0) = 0

Fibonacci(1) = 1

Fibonacci(*n*) = Fibonacci(*n* - 2) + Fibonacci(*n* - 1)

Example: fibonacc.c

```

function Fibonacci ( n )
{
    if ( n is less than or equal to 1 ) then
        return n
    else
        return Fibonacci ( n - 2 ) + Fibonacci ( n - 1 )
}

/* Compute the n-th Fibonacci number,
when=0,1,2,... */

long fib ( long n )
{
    if ( n <= 1 )
        return n ;
    else
        return fib( n - 2 ) + fib( n - 1 ) ;
}
    
```

Example: Computation of fib(4)

```

long fib ( long 4 )
{
    if ( 4 <= 1 )
        return n ;
    else
        return fib( 4 - 2 ) + fib( 4 - 1 ) ;
}
    
```

**Example:** Computation of **fib(4)**

```

long fib ( long 2 )
{
  if ( 2 <= 1 )
    return n ;
  else
    return fib( 2 - 2 ) + fib( 2 - 1 );
}
    
```

55

**Example:** Computation of **fib(4)**

```

long fib ( long 0 )
{
  if ( 0 <= 1 )
    return 0 ;
  else
    return fib( n - 2 ) + fib( n - 1 );
}
    
```

56

**Example:** Computation of **fib(4)**

```

long fib ( long 0 )
{
  if ( 0 <= 1 )
    return 0 ;
  else
    return fib( n - 2 ) + fib( n - 1 );
}
    
```

57

**Example:** Computation of **fib(4)**

```

long fib ( long 2 )
{
  if ( 2 <= 1 )
    return n ;
  else
    return 0 + fib( 2 - 1 );
}
    
```

58

**Example:** Computation of **fib(4)**

```

long fib ( long 1 )
{
  if ( 1 <= 1 )
    return 1 ;
  else
    return fib( n - 2 ) + fib( n - 1 );
}
    
```

59

**Example:** Computation of **fib(4)**

```

long fib ( long 1 )
{
  if ( 1 <= 1 )
    return 1 ;
  else
    return fib( n - 2 ) + fib( n - 1 );
}
    
```

60

**Example:** Computation of **fib(4)**

```

long fib ( long 2 )
{
  if ( 2 <= 1 )
    return n ;
  else
    return 0 + 1 ;
}
    
```

Diagram showing the computation of fib(4) as fib(2) + fib(3). fib(2) is further broken down into 0 + 1.

61

**Example:** Computation of **fib(4)**

```

long fib ( long 2 )
{
  if ( 2 <= 1 )
    return n ;
  else
    return 0 + 1 ;
}
    
```

Diagram showing the computation of fib(4) as fib(2) + fib(3). fib(2) is highlighted in blue and returns 1.

62

**Example:** Computation of **fib(4)**

```

long fib ( long 4 )
{
  if ( 4 <= 1 )
    return n ;
  else
    return 1 + fib( 4 - 1 ) ;
}
    
```

Diagram showing the computation of fib(4) as fib(2) + fib(3). fib(2) is highlighted in blue and returns 1.

63

**Example:** Computation of **fib(4)**

```

long fib ( long 3 )
{
  if ( 3 <= 1 )
    return n ;
  else
    return fib( 3 - 2 ) + fib( 3 - 1 ) ;
}
    
```

Diagram showing the computation of fib(4) as fib(2) + fib(3). fib(3) is further broken down into fib(1) + fib(2).

64

**Example:** Computation of **fib(4)**

```

long fib ( long 1 )
{
  if ( 1 <= 1 )
    return 1 ;
  else
    return fib( n - 2 ) + fib( n - 1 ) ;
}
    
```

Diagram showing the computation of fib(4) as fib(2) + fib(3). fib(3) is further broken down into fib(1) + fib(2).

65

**Example:** Computation of **fib(4)**

```

long fib ( long 1 )
{
  if ( 1 <= 1 )
    return 1 ;
  else
    return fib( n - 2 ) + fib( n - 1 ) ;
}
    
```

Diagram showing the computation of fib(4) as fib(2) + fib(3). fib(3) is further broken down into fib(1) + fib(2). fib(1) is highlighted in blue and returns 1.

66

**Example: Computation of fib(4)**

```

long fib ( long 3 )
{
  if ( 3 <= 1 )
    return n ;
  else
    return 1 + fib( 3 - 1 );
}
    
```

67

**Example: Computation of fib(4)**

```

long fib ( long 2 )
{
  if ( 2 <= 1 )
    return n ;
  else
    return fib( 2 - 2 ) + fib( 2 - 1 );
}
    
```

68

**Example: Computation of fib(4)**

```

long fib ( long 0 )
{
  if ( 0 <= 1 )
    return 0 ;
  else
    return fib( n - 2 ) + fib( n - 1 );
}
    
```

69

**Example: Computation of fib(4)**

```

long fib ( long 0 )
{
  if ( 0 <= 1 )
    return 0 ;
  else
    return fib( n - 2 ) + fib( n - 1 );
}
    
```

70

**Example: Computation of fib(4)**

```

long fib ( long 2 )
{
  if ( 2 <= 1 )
    return n ;
  else
    return 0 + fib( 2 - 1 );
}
    
```

71

**Example: Computation of fib(4)**

```

long fib ( long 1 )
{
  if ( 1 <= 1 )
    return 1 ;
  else
    return fib( n - 2 ) + fib( n - 1 );
}
    
```

72

**Example: Computation of fib(4)**

```

long fib ( long 1 )
{
  if ( 1 <= 1 )
    return 1 ;
  else
    return fib( n - 2 ) + fib( n - 1 ) ;
}
    
```

73

**Example: Computation of fib(4)**

```

long fib ( long 2 )
{
  if ( 2 <= 1 )
    return n ;
  else
    return 0 + 1 ;
}
    
```

74

**Example: Computation of fib(4)**

```

long fib ( long 2 )
{
  if ( 2 <= 1 )
    return n ;
  else
    return 0 + 1 ;
}
    
```

75

**Example: Computation of fib(4)**

```

long fib ( long 3 )
{
  if ( 3 <= 1 )
    return n ;
  else
    return 1 + 1 ;
}
    
```

76

**Example: Computation of fib(4)**

```

long fib ( long 3 )
{
  if ( 3 <= 1 )
    return n ;
  else
    return 1 + 1 ;
}
    
```

77

**Example: Computation of fib(4)**

```

long fib ( long 4 )
{
  if ( 4 <= 1 )
    return n ;
  else
    return 1 + 2 ;
}
    
```

78

*Example:* Computation of **fib(4)**

```

long fib ( long 4 )
{
  if ( 4 <= 1 )
    return n ;
  else
    return 1 + 2 ;
}
    
```

79

*Example:* Computation of **fib(4)**

Thus, **fib(4)** returns the value 3.

80

*Example:* fibonacc.c

Sample **main()** for testing the **fib()** function:

```

int main(void)
{
  long number;

  printf("Enter number: ");
  scanf("%ld", &number);

  printf("Fibonacci(%ld) = %ld\n",
         number, fib(number));

  return 0;
}
    
```

81

## Reading

- King Chapter 9  
Section 9.6
- D&D Chapter 5  
Sections 5.13 to 5.14
- Kernighan & Ritchie Chapter 4  
Section 4.10

82