

# Welcome to ICS 111 Lab

Section 3 and 4

Week 8

**\*\*Read through HW7 Assignment\*\***

# Tuesday Outline

- Recursion
- Hexadecimal
- HW 7

# Recursion:

- A recursion (recursive function) is a function that calls itself.
- To solve a recursive function:
  - 1. Divide the problem into subproblems
  - 2. Specify base case to stop the recursion.
- Structure (for factorial example):

```
function() {  
    if() { //base case (2)  
        ... return}  
  
    else{... return} //recursive call (1)  
}
```

# Recursion: Factorial Example

$$5! = 5 * 4 * 3 * 2 * 1$$

$$4! = 4 * 3 * 2 * 1$$

$$3! = 3 * 2 * 1$$

$$2! = 2 * 1$$

$$1! = 1$$

# Recursion: Factorial Pattern Example

$$5! = 5 * 4 * 3 * 2 * 1$$

$$4! = 4 * 3 * 2 * 1$$

$$3! = 3 * 2 * 1$$

$$2! = 2 * 1$$

$$1! = 1$$

## Recursion: Factorial Subproblem

$$5! = 5 * 4 * 3 * 2 * 1$$

The diagram illustrates the recursive calculation of 5!. The expression  $5! = 5 * 4 * 3 * 2 * 1$  is shown. The number 5 is highlighted in purple. A red asterisk is placed between the 5 and the 4. Below the 5, another purple 5 is shown with an upward-pointing arrow. A bracket underneath the terms 4, 3, 2, and 1 groups them together, with a 4! written below the bracket. A red asterisk is also placed between the purple 5 and the bracketed group.

$$n! = n * (n-1)!$$

Recursion: Where is the base case?

$$5! = 5 * 4 * 3 * 2 * 1$$

$$4! = 4 * 3 * 2 * 1$$

$$3! = 3 * 2 * 1$$

$$2! = 2 * 1$$

$$1! = 1$$

## Recursion: base case

$$n! = n * \cancel{(n-1)!}$$

$$1! = 1$$

Note: Base case will be evaluated in the if() condition, to end the recursion.



## Recursion: Factorial Subproblem

$$5! = 5 * 4 * 3 * 2 * 1$$

The diagram illustrates the recursive calculation of a factorial. The expression  $5! = 5 * 4 * 3 * 2 * 1$  is shown. The number 5 is highlighted in purple. A red asterisk is placed between the 5 and the 4. Below the 5, another purple 5 is shown with an upward-pointing arrow. A bracket groups the terms 4, 3, 2, and 1, with a 4! written below it. A red asterisk is also placed between the purple 5 and the 4!

$$n! = n * (n-1)!$$

# Factorial:

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

**Conditional** (points to `n==1`)

**Base Case** (points to the `if` statement)

**Recursive Call** (points to `factorial(n-1)`)

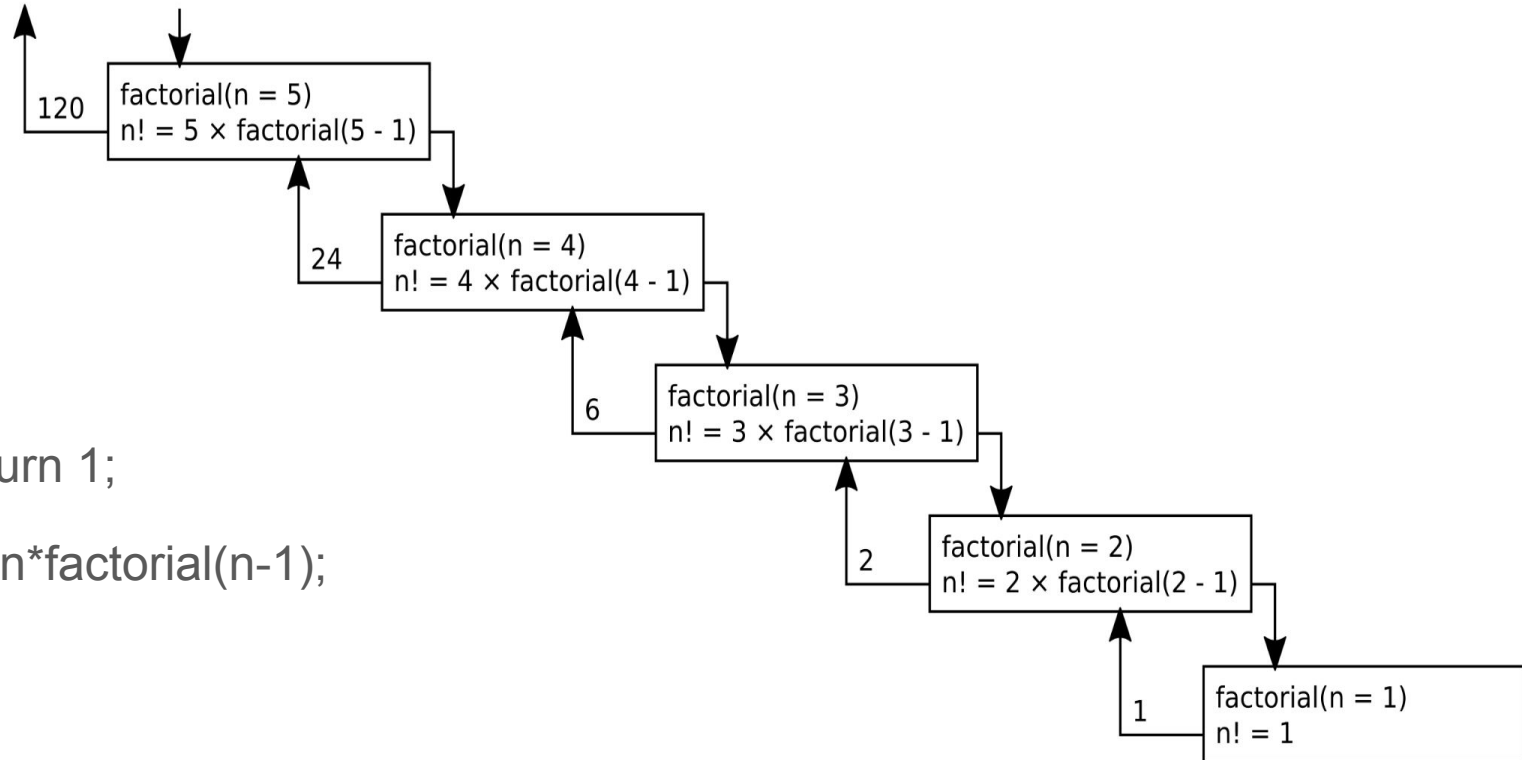
**Sub problem** (points to `n`)

**Modified input** (points to `n-1`)

Base Case: Return without making recursive call and stop function.

Recursive Call: Function calls itself.

# Factorial:



`factorial(n)`

`if(n==1) return 1;`

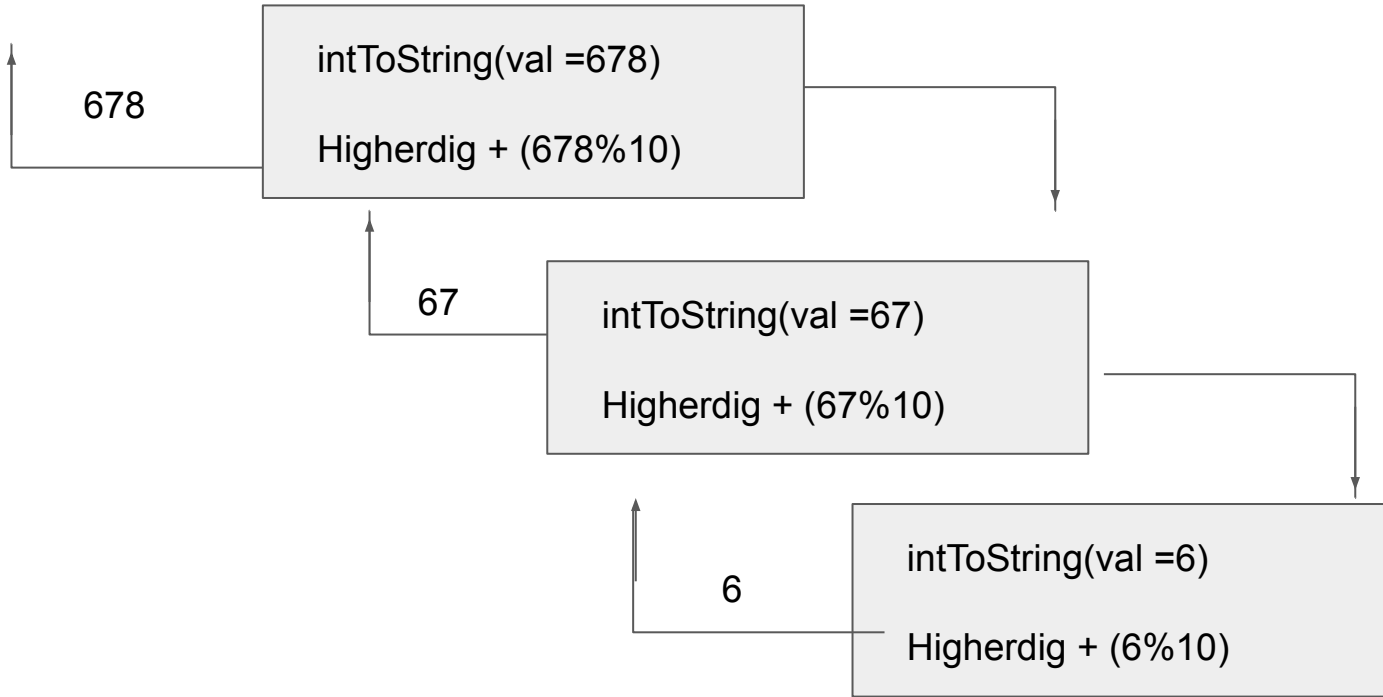
`else return n*factorial(n-1);`

# Factorial: intToString()

```
intToString(val)
String higherdigits = "";
if(val>=10) {
    higherdigits=intToString(val/10);
}
return higherdigits + (val%10); }
```

Activity: Draw diagram of recursive function/  
Describe what is happening.

# Factorial: intToString() : Diagram Solution



# Hexadecimal:

- 0-9 : Same as decimal (unit)
- 10-15: A-F (unit)
- 16...

**16 = 1 0**  
16 Unit

**17 = 1 1**  
16 Unit

	4 bit	bits
Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	0001 0000

# Hexadecimal: Resources

See chart: <https://kb.iu.edu/d/afdl>

Counting in Hex: [https://www.electronics-tutorials.ws/binary/bin\\_3.html](https://www.electronics-tutorials.ws/binary/bin_3.html)

# Week 8: Resources

- HW 7:  
<http://www2.hawaii.edu/~esb/2022spring.ics111/hw07.html>
- Neso Academy -Recursive Functions: <https://www.youtube.com/watch?v=ggk7HbcnLG8>
- Geeks for Geeks - Recursive Function types:  
<https://www.geeksforgeeks.org/types-of-recursions/>
- Textbook PDF:  
<http://bedford-computing.co.uk/learning/wp-content/uploads/2015/09/Java-for-Everyone-Late-Objects.pdf>