

CS 161

Intro to CS I

Finish Recursion/Begin Memory Model



Odds and Ends

- Assignment 5 posted
- Assignment 4 demo this week
- 1 credit hour of lecture/course is 3 hours outside the course
- Poor planning on your part does not constitute an emergency on mine!
- KISS!

Iterative Factorial



factorial(0) = 1;

factorial(n) = $n * n-1 * n-2 * \dots * n-(n-1) * 1$;

```
long factorial(int n) {  
    long fact;  
    if(n==0)  
        fact=1;  
    else  
        for(fact=n; n > 1; n--)  
            fact=fact*(n-1);  
    return fact;  
}
```



Recursive Factorial

```
factorial(0) = 1; base case  
factorial(n) = n * factorial(n-1);
```

```
long factorial(int n) {  
    if (n == 0) // Base case  
        return 1;  
    else  
        return n * factorial(n - 1); // Recursive call  
}
```

*getting you
closer to
base case*

Computing Factorial Iteratively



Oregon State University
College of Engineering

factorial(4)

```
factorial(0) = 1;
```

```
factorial(n) = n*(n-1)*...*2*1;
```

Computing Factorial Iteratively



Oregon State University
College of Engineering

$$\text{factorial}(4) = 4 * 3$$

```
factorial(0) = 1;  
factorial(n) = n*(n-1)*...*2*1;
```

Computing Factorial Iteratively



Oregon State University
College of Engineering

$$\begin{aligned} \text{factorial}(4) &= 4 * 3 \\ &= 12 * 2 \end{aligned}$$

```
factorial(0) = 1;  
factorial(n) = n*(n-1)*...*2*1;
```

Computing Factorial Iteratively



Oregon State University
College of Engineering

$$\begin{aligned}\text{factorial}(4) &= 4 * 3 \\ &= \underline{12} * 2 \\ &= 24 * 1\end{aligned}$$

```
factorial(0) = 1;  
factorial(n) = n*(n-1)*...*2*1;
```


Computing Factorial Iteratively



Oregon State University
College of Engineering

$$\begin{aligned}\text{factorial}(4) &= 4 * 3 \\ &= 12 * 2 \\ &= 24 * 1 \\ &= 24\end{aligned}$$

```
factorial(0) = 1;  
factorial(n) = n*(n-1)*...*2*1;
```

Computing Factorial Recursively



Oregon State University
College of Engineering

factorial(4)

```
factorial(0) = 1;
```

```
factorial(n) = n*factorial(n-1);
```

Computing Factorial Recursively



Oregon State University
College of Engineering

```
factorial(0) = 1;  
factorial(n) = n * factorial(n-1);
```

$$\text{factorial}(4) = 4 * \text{factorial}(3)$$

Computing Factorial Recursively



Oregon State University
College of Engineering

```
factorial(0) = 1;  
factorial(n) = n * factorial(n-1);
```

$$\begin{aligned} \text{factorial}(4) &= 4 * \text{factorial}(3) \\ &= \underline{4} * (3 * \text{factorial}(2)) \end{aligned}$$

Computing Factorial Recursively



Oregon State University
College of Engineering

```
factorial(0) = 1;  
factorial(n) = n * factorial(n-1);
```

$$\begin{aligned}\text{factorial}(4) &= 4 * \text{factorial}(3) \\ &= 4 * (3 * \text{factorial}(2)) \\ &= 4 * (3 * (2 * \text{factorial}(1)))\end{aligned}$$

Computing Factorial Recursively



Oregon State University
College of Engineering

```
factorial(0) = 1;  
factorial(n) = n * factorial(n-1);
```

$$\begin{aligned} \text{factorial}(4) &= 4 * \text{factorial}(3) \\ &= 4 * (3 * \text{factorial}(2)) \\ &= 4 * (3 * (2 * \text{factorial}(1))) \\ &= 4 * (3 * (2 * (1 * \text{factorial}(0)))) \end{aligned}$$

Computing Factorial Recursively



Oregon State University
College of Engineering

```
factorial(0) = 1;  
factorial(n) = n * factorial(n-1);
```

$$\begin{aligned}\text{factorial}(4) &= 4 * \text{factorial}(3) \\ &= 4 * (3 * \text{factorial}(2)) \\ &= 4 * (3 * (2 * \text{factorial}(1))) \\ &= 4 * (3 * (2 * (1 * \text{factorial}(0)))) \\ &= 4 * (3 * (2 * (1 * 1)))\end{aligned}$$

Computing Factorial Recursively



Oregon State University
College of Engineering

```
factorial(0) = 1;  
factorial(n) = n * factorial(n-1);
```

$$\begin{aligned}\text{factorial}(4) &= 4 * \text{factorial}(3) \\ &= 4 * (3 * \text{factorial}(2)) \\ &= 4 * (3 * (2 * \text{factorial}(1))) \\ &= 4 * (3 * (2 * (1 * \text{factorial}(0)))) \\ &= 4 * (3 * (2 * (1 * 1))) \\ &= 4 * (3 * (2 * 1))\end{aligned}$$

Computing Factorial Recursively



Oregon State University
College of Engineering

```
factorial(0) = 1;  
factorial(n) = n * factorial(n-1);
```

$$\begin{aligned} \text{factorial}(4) &= 4 * \text{factorial}(3) \\ &= 4 * (3 * \text{factorial}(2)) \\ &= 4 * (3 * (2 * \text{factorial}(1))) \\ &= 4 * (3 * (2 * (1 * \text{factorial}(0)))) \\ &= 4 * (3 * (2 * (1 * 1))) \\ &= 4 * (3 * (2 * 1)) \\ &= 4 * (3 * 2) \end{aligned}$$

Computing Factorial Recursively



Oregon State University
College of Engineering

```
factorial(0) = 1;  
factorial(n) = n * factorial(n-1);
```

$$\begin{aligned} \text{factorial}(4) &= 4 * \text{factorial}(3) \\ &= 4 * (3 * \text{factorial}(2)) \\ &= 4 * (3 * (2 * \text{factorial}(1))) \\ &= 4 * (3 * (2 * (1 * \text{factorial}(0)))) \\ &= 4 * (3 * (2 * (1 * 1))) \\ &= 4 * (3 * (2 * 1)) \\ &= 4 * (3 * 2) \\ &= 4 * 6 \end{aligned}$$

Computing Factorial Recursively



Oregon State University
College of Engineering

`factorial(0) = 1;`

`factorial(n) = n*factorial(n-1);`

$$\begin{aligned} \text{factorial}(4) &= 4 * \text{factorial}(3) \\ &= 4 * (3 * \text{factorial}(2)) \\ &= 4 * (3 * (2 * \text{factorial}(1))) \\ &= 4 * (3 * (2 * (1 * \text{factorial}(0)))) \\ &= 4 * (3 * (2 * (1 * 1))) \\ &= 4 * (3 * (2 * 1)) \\ &= 4 * (3 * 2) \\ &= 4 * 6 \\ &= 24 \end{aligned}$$

Differences



Oregon State University
College of Engineering

- Pros
 - Readability
- Cons
 - Efficiency
 - Memory

You have to have shallow depth

not too many recursive calls.

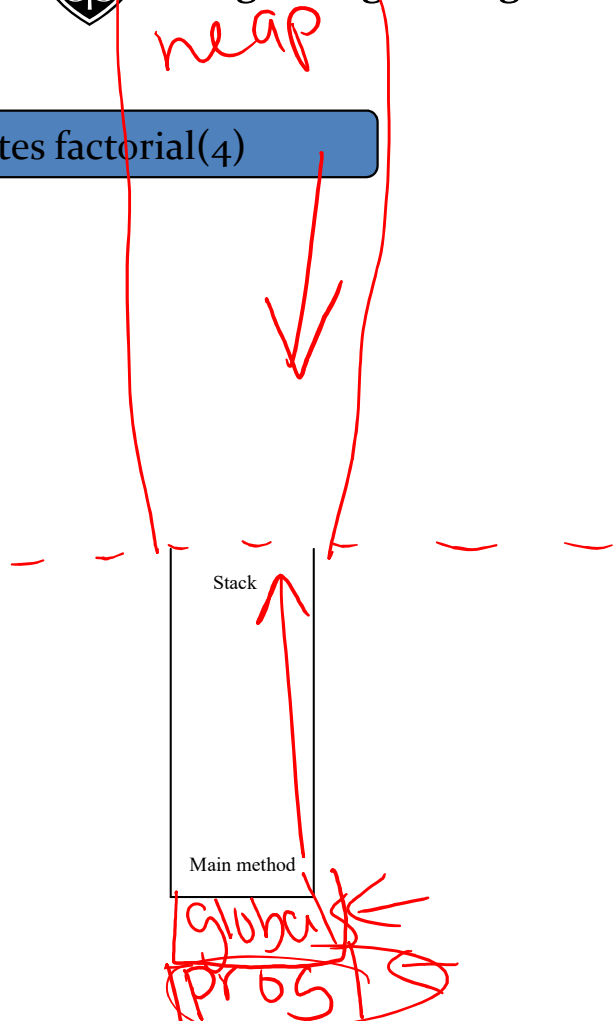
Recursive Factorial



Oregon State University
College of Engineering

factorial(4)

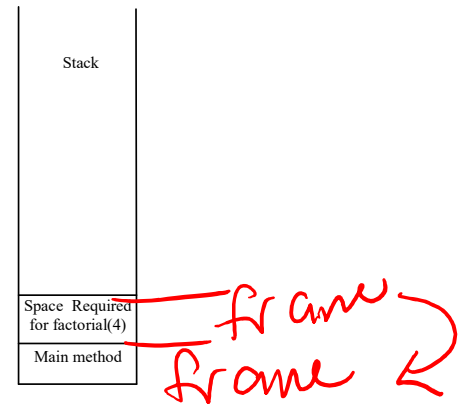
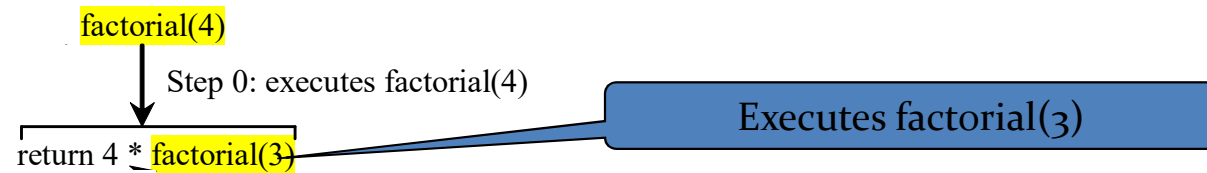
Executes factorial(4)



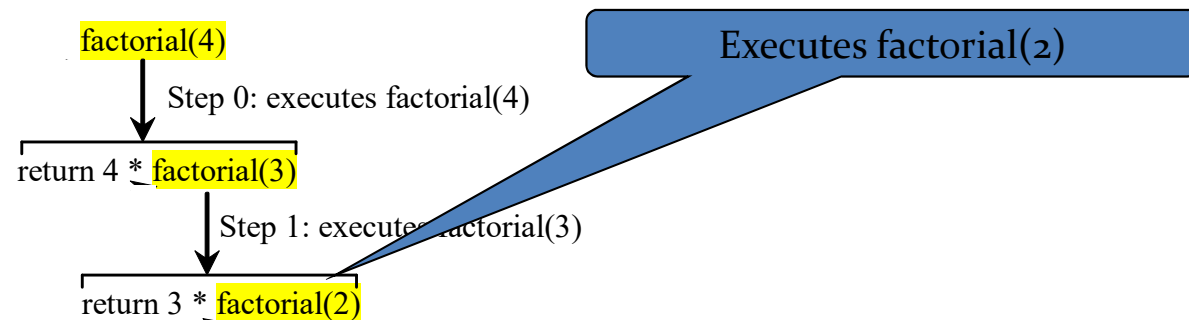
Recursive Factorial



Oregon State University
College of Engineering

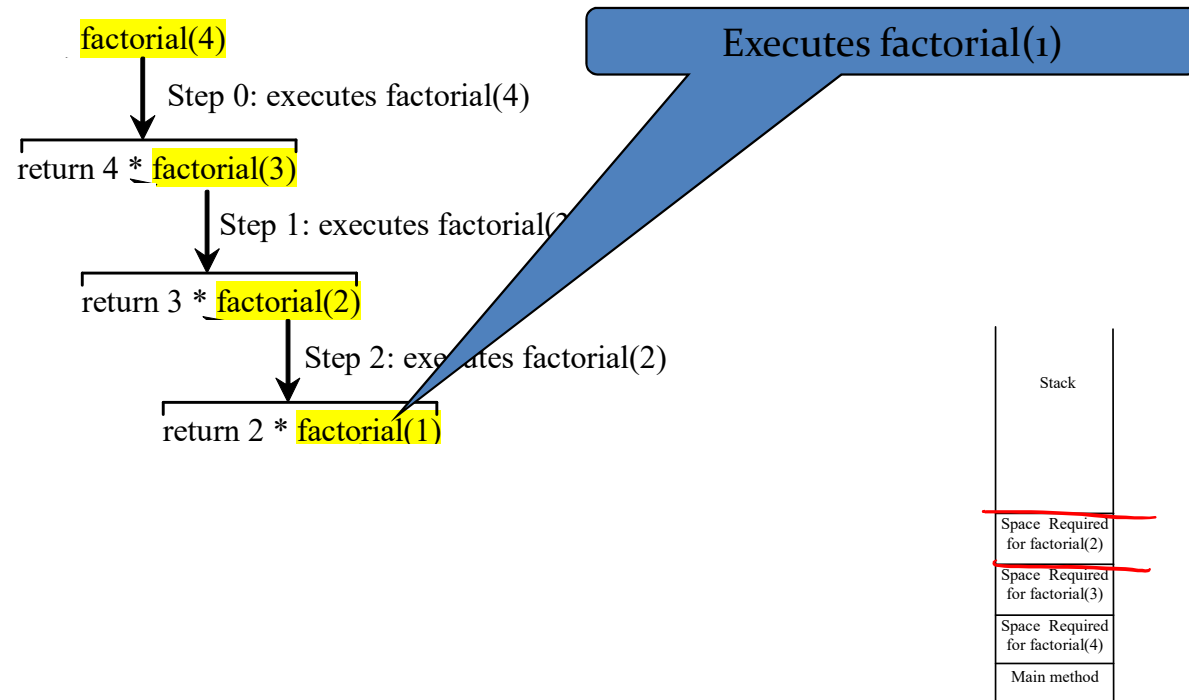


Recursive Factorial

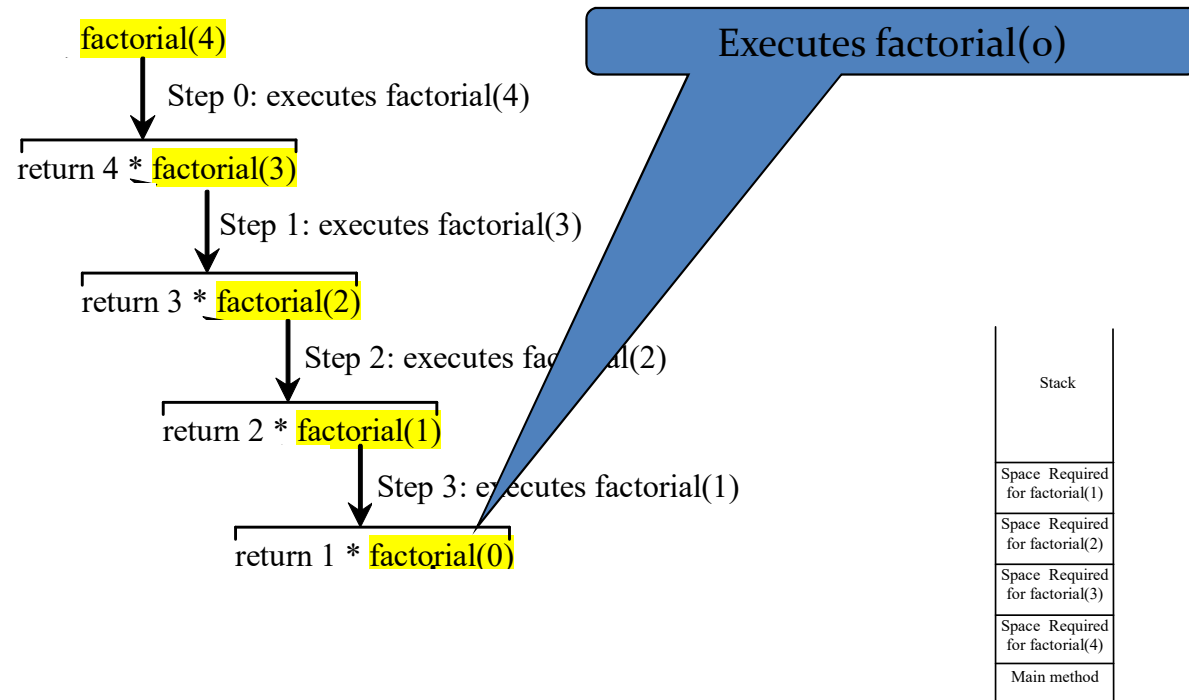


Stack
Space Required for factorial(3)
Space Required for factorial(4)
Main method

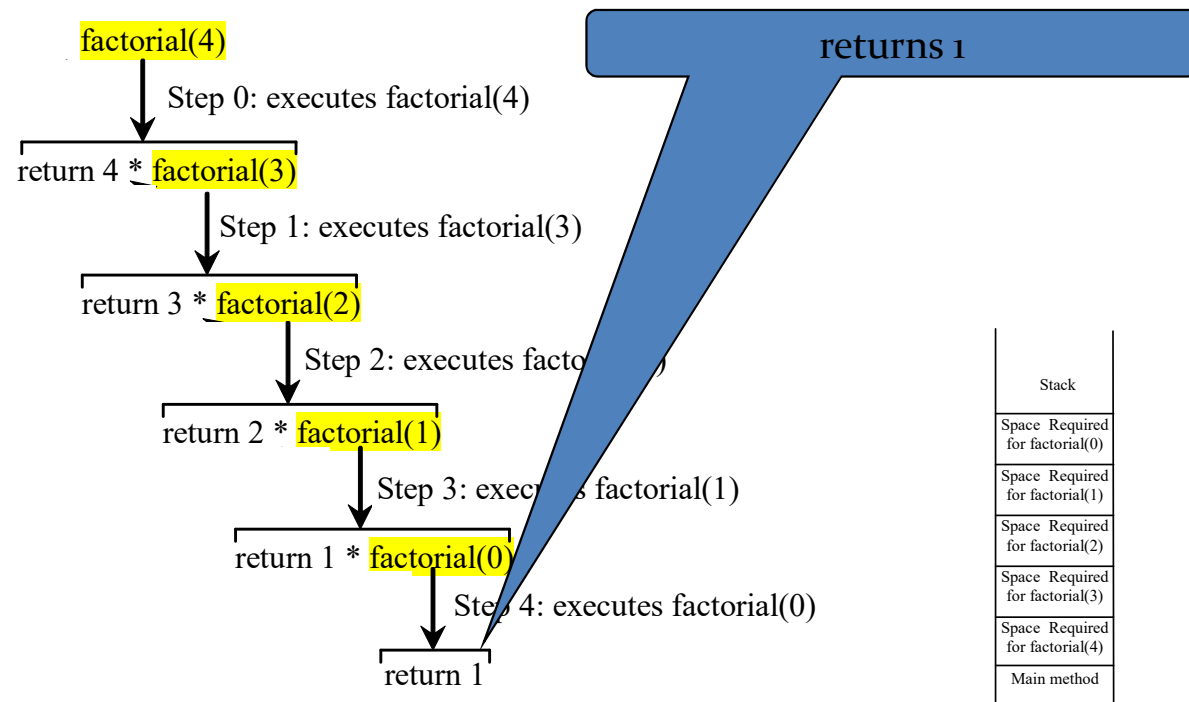
Recursive Factorial



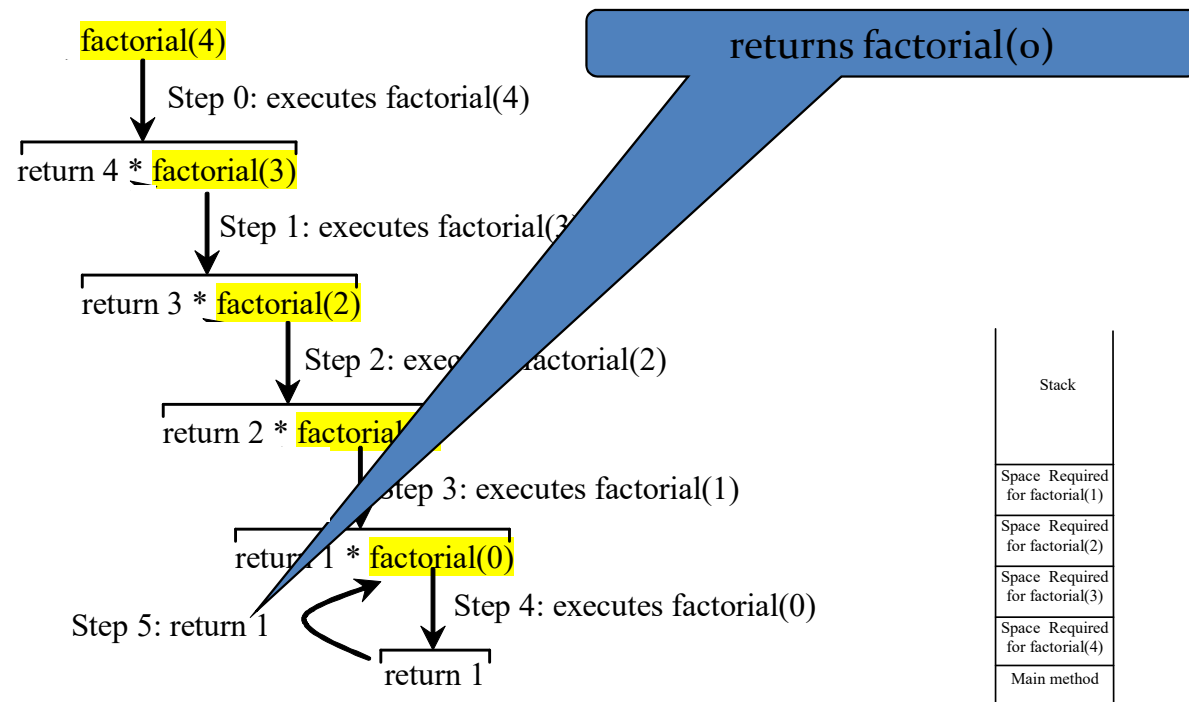
Recursive Factorial



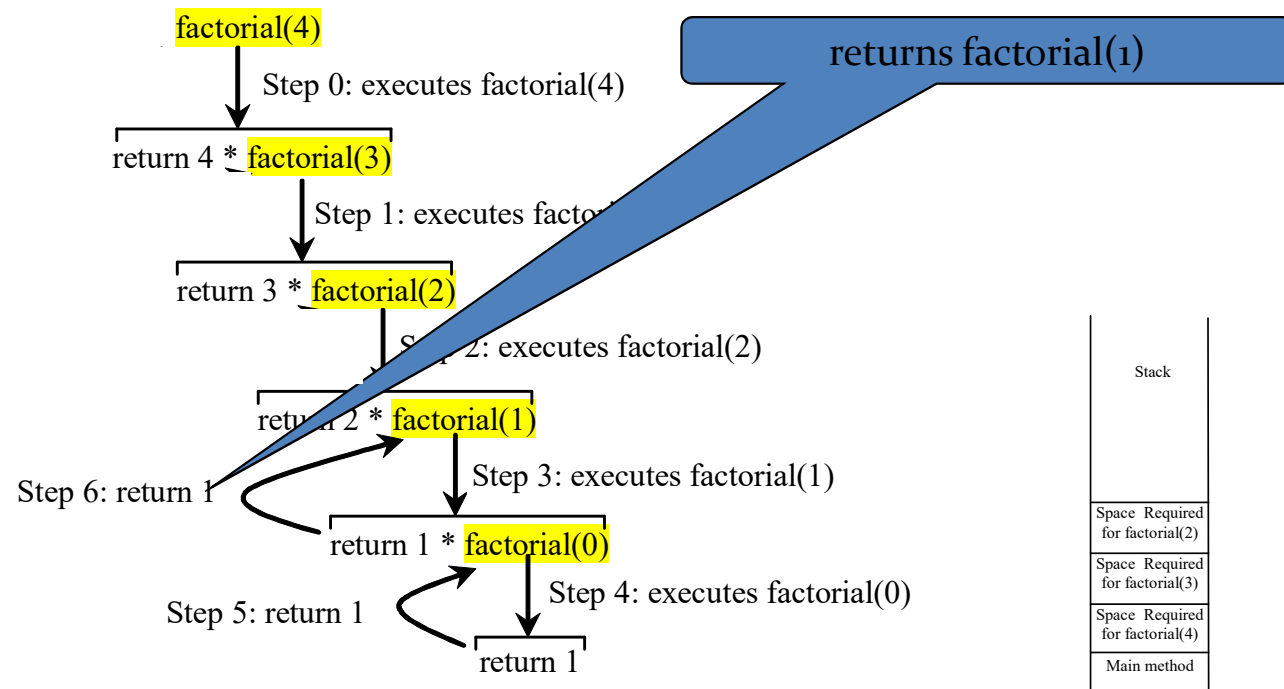
Recursive Factorial



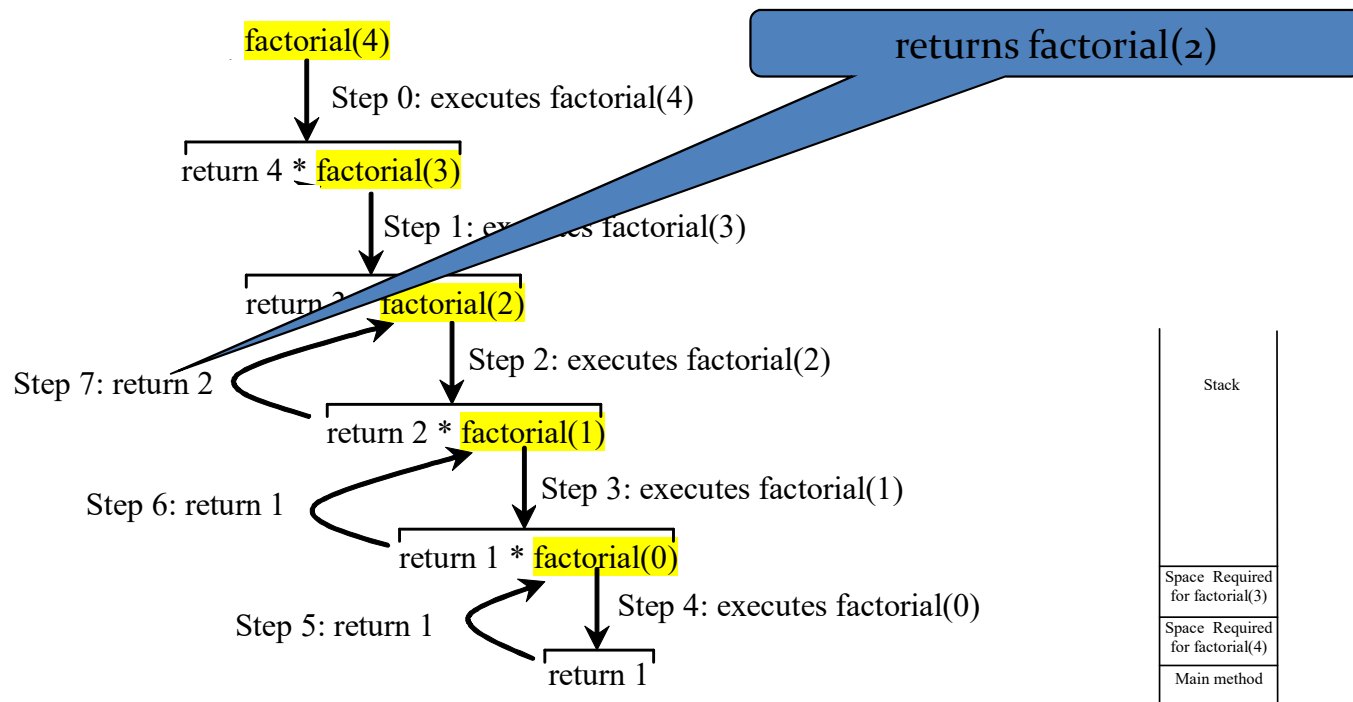
Recursive Factorial



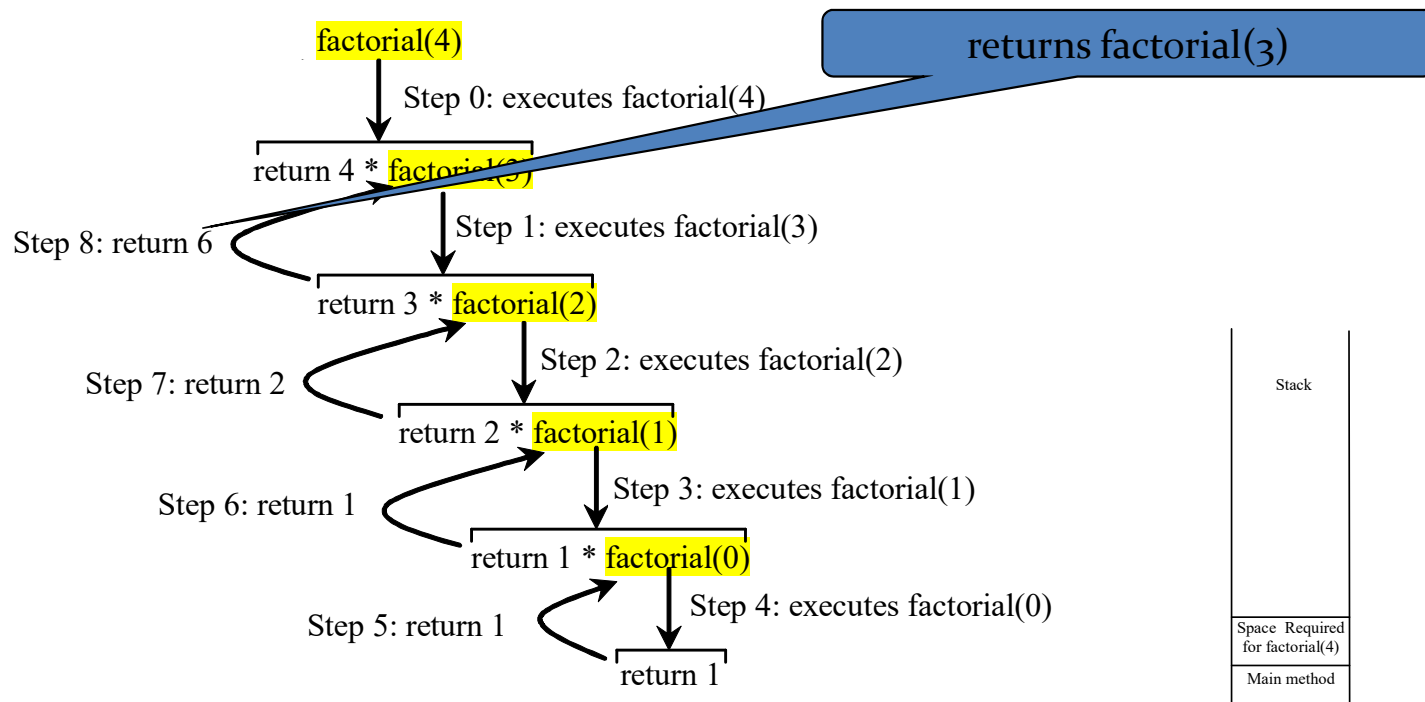
Recursive Factorial



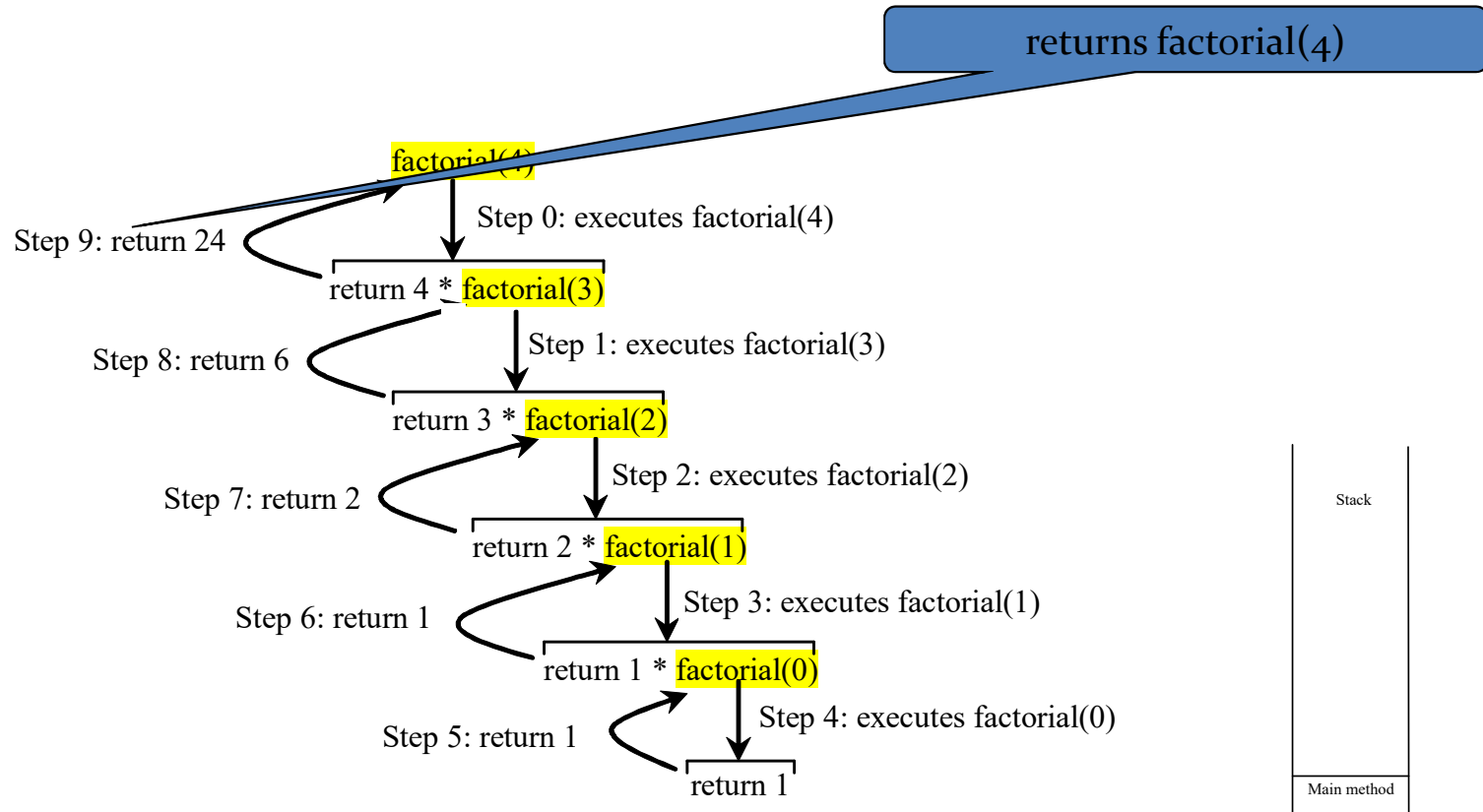
Recursive Factorial



Recursive Factorial



Recursive Factorial



In-class Exercise #4



Oregon State University
College of Engineering

- ~~Get into groups of 4 – 5.~~
- Write your own recursive *int pwr()* function that takes two integers as arguments and returns the integer result.
 - What does the function prototype look like?
 - Now, write the function definition...


```
4. ENGR
Re-attach Fullscreen Stay on top Duplicate
1 #include <iostream>
2
3 using namespace std;
4
5 int pwr(int base, int exp) {
6     int result=1;
7
8     for(int i=0; i<exp; i++) {
9         result=result*base;
10    }
11
12    return result;
13 }
14 int pwr_r(int base, int exp) {
15     if(exp==0) //base case
16         return 1;
17     else
18         return base*pwr_r(base, exp-1); //get us closer to base case
19 }
20
21 int main() {
22
23     cout << pwr(2,1000000) << endl;
24     cout << pwr_r(2,1000000) << endl; //too deep of recursion, blow stack
25
-- INSERT --
24,74 Top
```

sity
ering

Stack vs. Heap



- Static vs. Dynamic

created/memory is known about at compile time

created at runtime

```
int i;  
int x;
```

you control

you can't control

heap

stack

nameless

initial item created has a name



Static vs. Dynamic

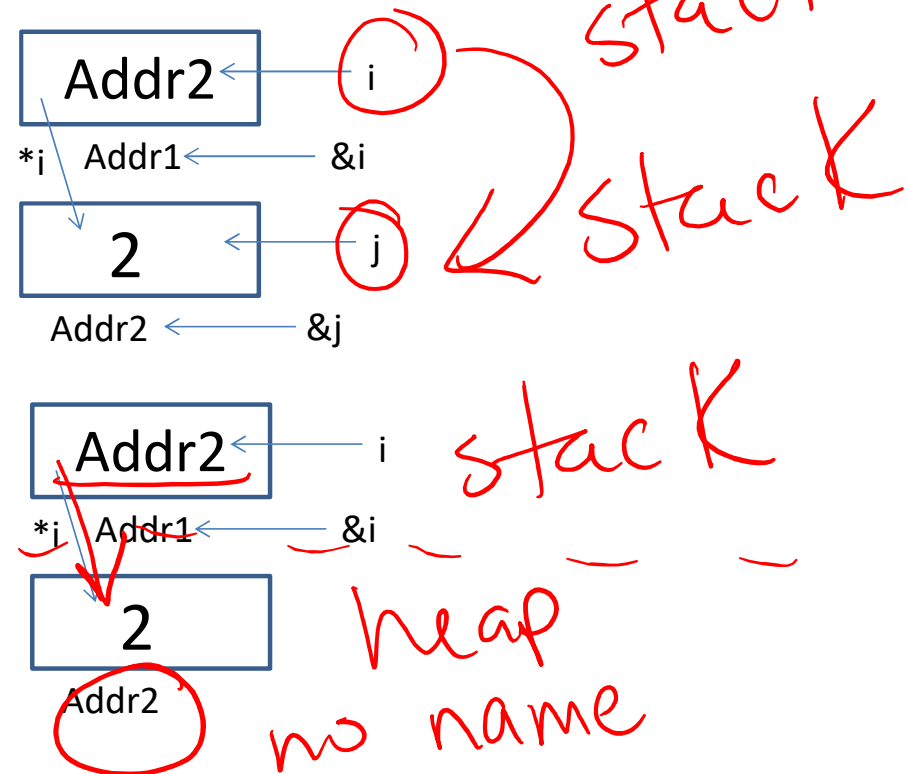
- Static Semantics
 - Assign address of variable

```
int *i, j=2;
i=&j;
```

- Dynamic Semantics
 - Create memory
 - Assign memory to pointer

```
int *i=NULL;
i=new int;
*i=2;
```

return addr.
what type
make mem on the heap





What About Memory Leaks?

- What happens here...

...

```
int main () {  
    int *i=NULL; //created in main function  
    while(1) {  
        i = new int;  
    }  
}
```

Fixing Memory Leaks...



- What happens here...

```
...  
int main () {  
    int *i=NULL;    //created in main function  
    while(1) {  
        i = new int;  
        delete i; //free memory that i points to, preventing mem leaks  
    }  
}
```

Handwritten red annotations:
- A red arrow points from the word "count" to the pointer variable "i".
- The word "count" is written in red above the "delete i;" line.
- The "delete i;" line is underlined in red.