CS 161
Intro to CS I

Recursion

Odds and Ends

- Assignment 4 due Sunday
- Questions?

<cstdlib>

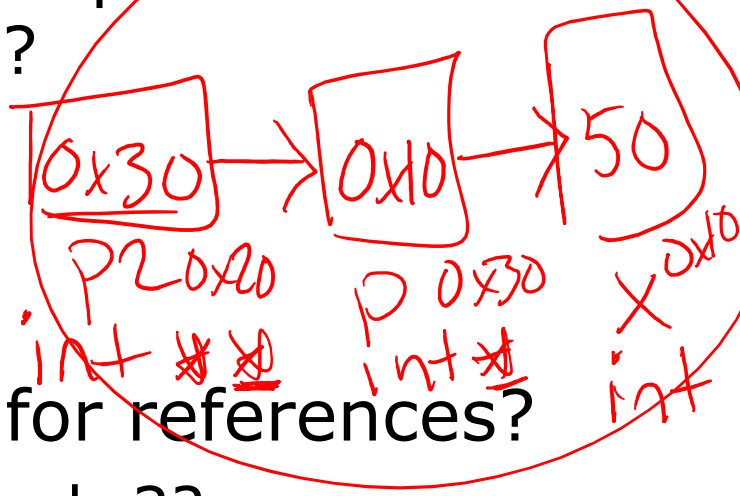
int x = atoi(s.c_str())

double y = atof(s.c_str())

C-style strings

Exercise Pointers vs. References

- What if you made a pointer (p2) that points to a pointer (p) that points to an int (x)?
  - What would the picture look like?
  - Write the code for this picture.




```
int **p2, *p, x=50;
```

- Can you make this same picture for references?
  - What if you had two references, r and r2?

```
p=&x;
p2=&p;
cout << *p; //50
cout << *p2; //0x10
cout << **p2; //50
```

Recursion = Repetition

Oregon State University
College of Engineering

- What is it?
  - Function that calls itself 1 or more times (directly or indirectly)
  - Has 1 or more base case for stopping
  - Inductive reasoning: general case must eventually be reduced to a base case

Example: Drawing Rectangles

- Iterative Solution:

```
void draw_rect(int i) {
        for( ; i > 0; i--){
           cout << "******" << endl;
           cout << "*      *" << endl;
           cout << "******" << endl;
        }
  }
```

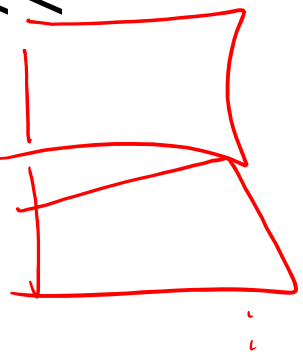
draw_rect(4);

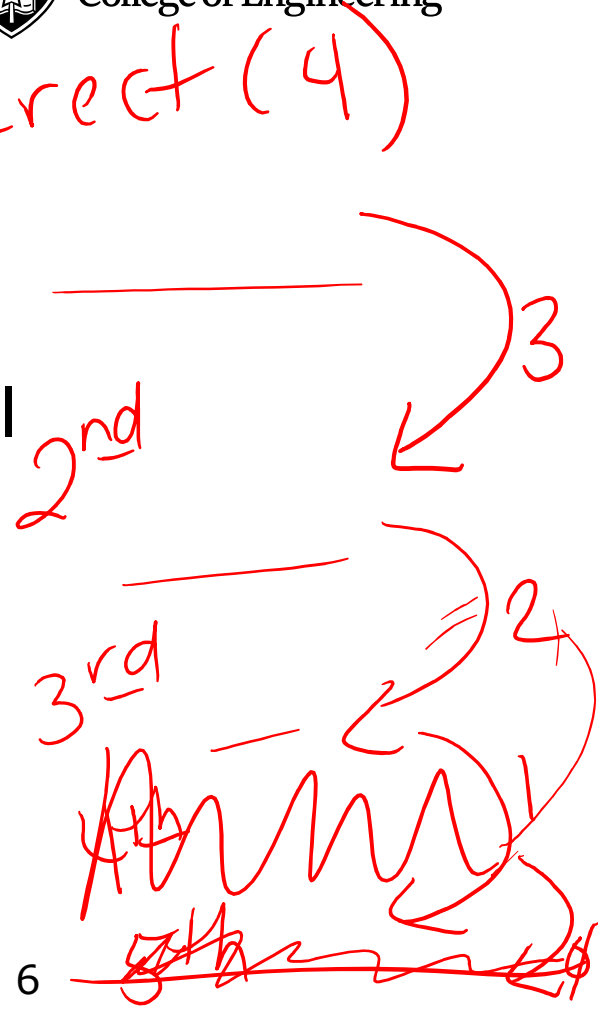
Example: Drawing Rectangles

Oregon State University
College of Engineering

- Recursive Solution

```
void draw_rect(int i) {
        if(i>0) {       //Base case
          draw_rect(--i);    //Recursive call
          cout << "******" << endl;
          cout << "*      *" << endl;
          cout << "******" << endl;
        }
  }
```



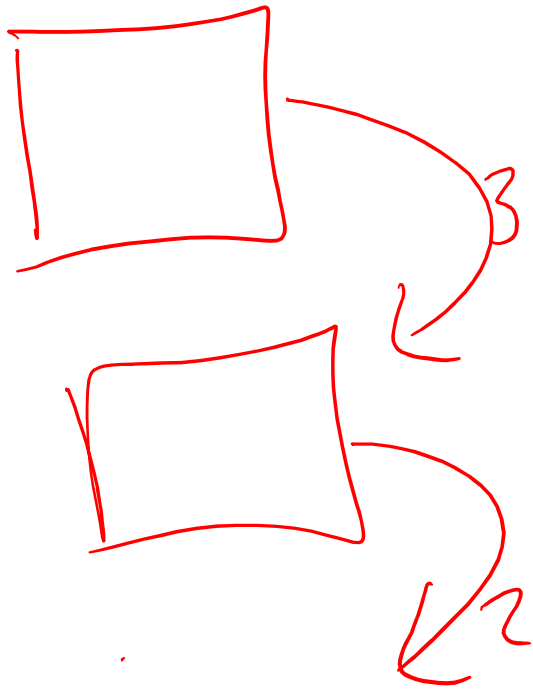


What is different when we call after?

Oregon State University
College of Engineering

- Recursive Solution

```
void draw_rect(int i) {
        if(i>0) {       //Base case
           cout << "******" << endl;
           cout << "*      *" << endl;
           cout << "******" << endl;
           draw_rect(--i);    //Recursive call
        }
  }
```



Example: Factorial

Oregon State University
College of Engineering

- Definition

  $0! = 1;$

  $n! = n * (n-1) * \ldots * (n-(n-1)) * 1 = n * (n-1)!\ ; n > 0$

Iterative Factorial

Oregon State University
College of Engineering

factorial(0) = 1;
factorial(n) = n*n-1*n-2*…*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

Oregon State University
College of Engineering

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

Computing Factorial Iteratively

Oregon State University
College of Engineering

factorial(0) = 1;
factorial(n) = n*(n-1)*…*2*1;

factorial(4)

Computing Factorial Iteratively

Oregon State University
College of Engineering

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

factorial(4) = 4 * 3

Computing Factorial Iteratively

Oregon State University
College of Engineering

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

factorial(4) = 4 * 3
= 12 * 2

Computing Factorial Iteratively

Oregon State University
College of Engineering

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

factorial(4) = 4 * 3
= 12 * 2
= 24 * 1

Computing Factorial Iteratively

Oregon State University
College of Engineering

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

factorial(4) = 4 * 3
= 12 * 2
= 24 * 1
= 24

Computing Factorial Recursively

Oregon State University
College of Engineering

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

factorial(4)

Computing Factorial Recursively

Oregon State University
College of Engineering

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

factorial(4) = 4 * factorial(3)

Computing Factorial Recursively

Oregon State University
College of Engineering

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

factorial(4) = 4 * factorial(3)
= 4 * (3 * factorial(2))

Computing Factorial Recursively

Oregon State University
College of Engineering

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

factorial(4) = 4 * factorial(3)
= 4 * (3 * factorial(2))
= 4 * (3 * (2 * factorial(1)))

Computing Factorial Recursively

Oregon State University
College of Engineering

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

factorial(4) = 4 * factorial(3)
= 4 * (3 * factorial(2))
= 4 * (3 * (2 * factorial(1)))
= 4 * (3 * (2 * (1 * factorial(0))))

Computing Factorial Recursively

Oregon State University
College of Engineering

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

factorial(4) = 4 * factorial(3)
= 4 * (3 * factorial(2))
= 4 * (3 * (2 * factorial(1)))
= 4 * (3 * (2 * (1 * factorial(0))))
= 4 * (3 * (2 * (1 *1)))

Computing Factorial Recursively

Oregon State University
College of Engineering

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

factorial(4) = 4 * factorial(3)
= 4 * (3 * factorial(2))
= 4 * (3 * (2 * factorial(1)))
= 4 * (3 * (2 * (1 * factorial(0))))
= 4 * (3 * (2 * (1 *1)))
= 4 * (3 * (2 * 1))

Computing Factorial Recursively

Oregon State University
College of Engineering

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

factorial(4) = 4 * factorial(3)
= 4 * (3 * factorial(2))
= 4 * (3 * (2 * factorial(1)))
= 4 * (3 * (2 * (1 * factorial(0))))
= 4 * (3 * (2 * (1 *1)))
= 4 * (3 * (2 * 1))
= 4 * (3 * 2)

Computing Factorial Recursively

Oregon State University
College of Engineering

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

factorial(4) = 4 * factorial(3)
= 4 * (3 * factorial(2))
= 4 * (3 * (2 * factorial(1)))
= 4 * (3 * (2 * (1 * factorial(0))))
= 4 * (3 * (2 * (1 *1)))
= 4 * (3 * (2 * 1))
= 4 * (3 * 2)
= 4 * 6

Computing Factorial Recursively

Oregon State University
College of Engineering

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

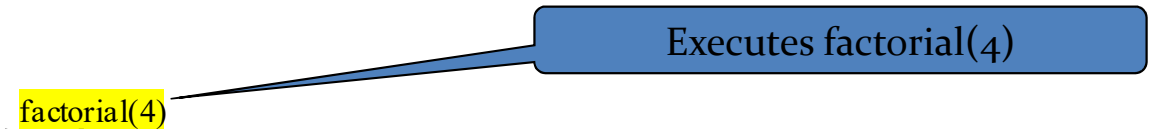
```
factorial(4) = 4 * factorial(3)
             = 4 * (3 * factorial(2))
             = 4 * ( 3 * (2 * factorial(1)))
             = 4 * ( 3 * ( 2 * (1 * factorial(0))))
             = 4 * ( 3 * ( 2 * (1 *1)))
             = 4 * ( 3 * ( 2 * 1))
             = 4 * (3 * 2)
             = 4 * 6
             = 24
```

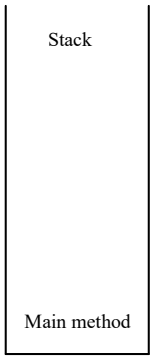
Differences

Oregon State University
College of Engineering

- Pros
  - Readability
- Cons
  - Efficiency
  - Memory

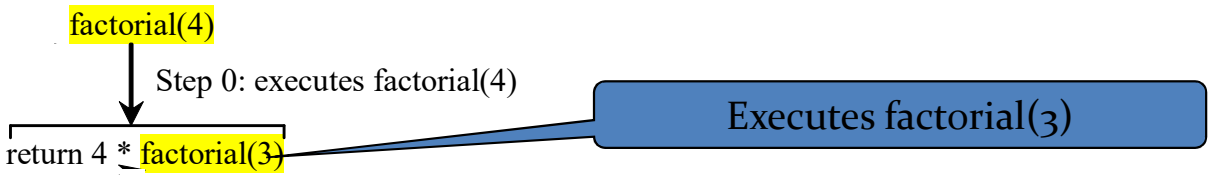
Recursive Factorial

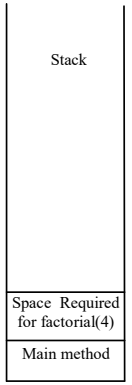




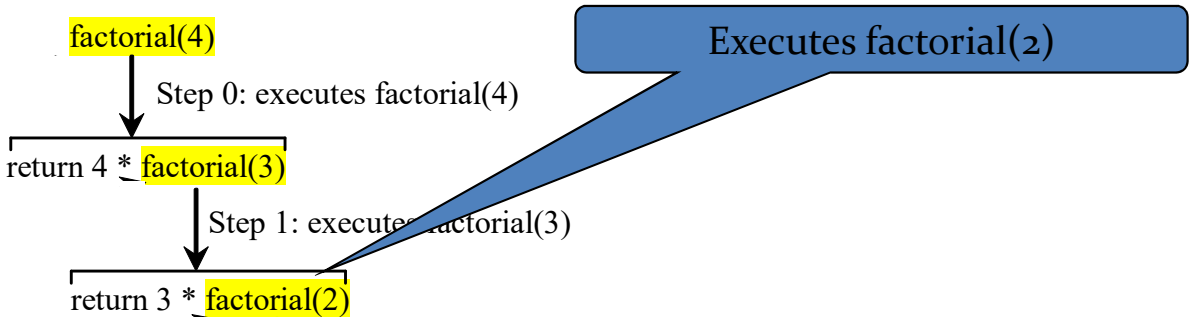
Recursive Factorial

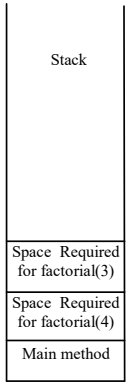
Oregon State University
College of Engineering





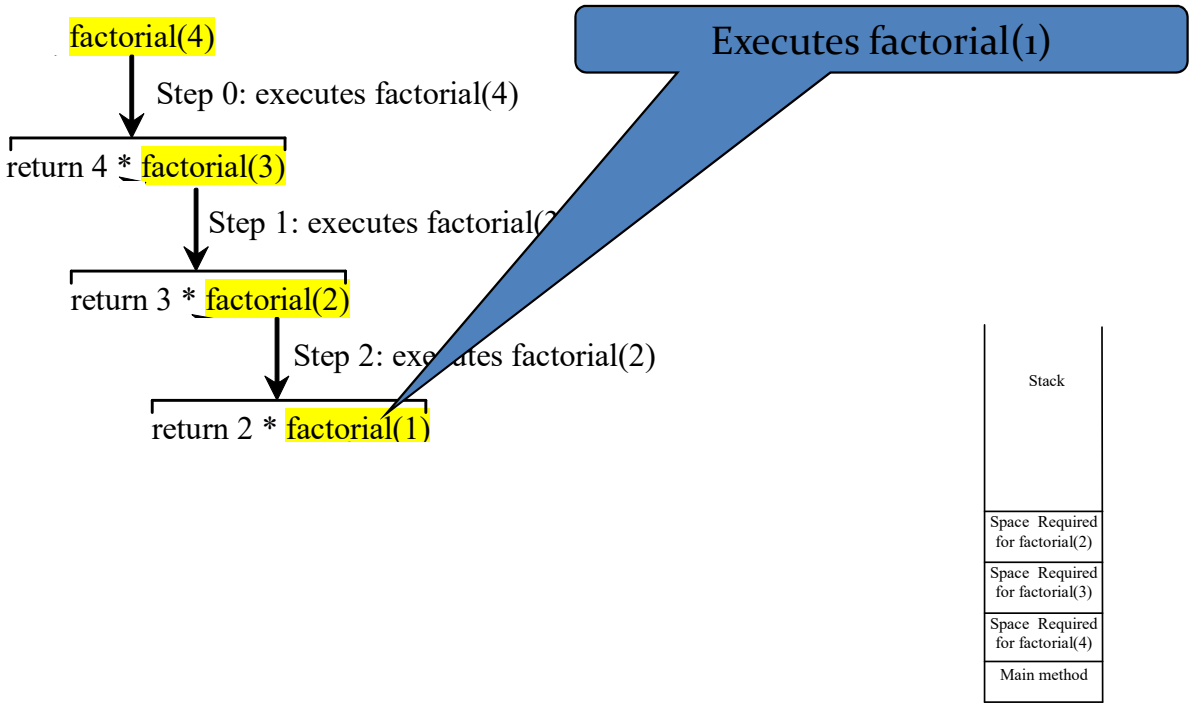
Recursive Factorial



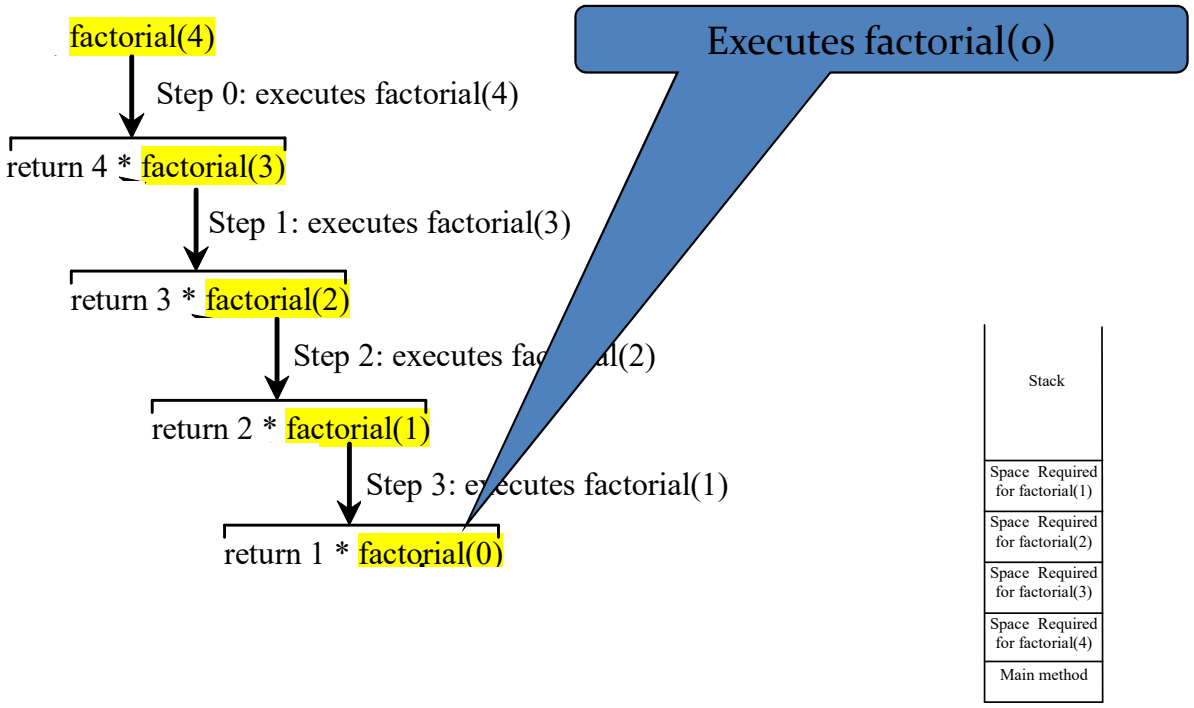


Recursive Factorial

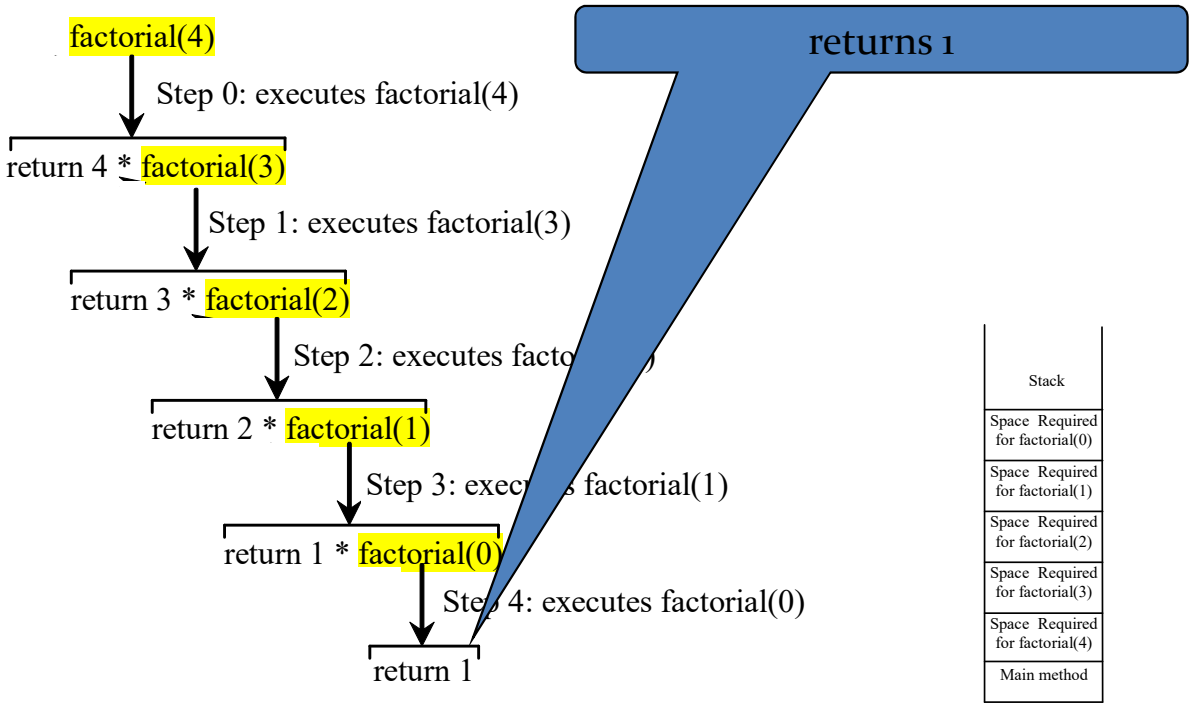
Oregon State University
College of Engineering



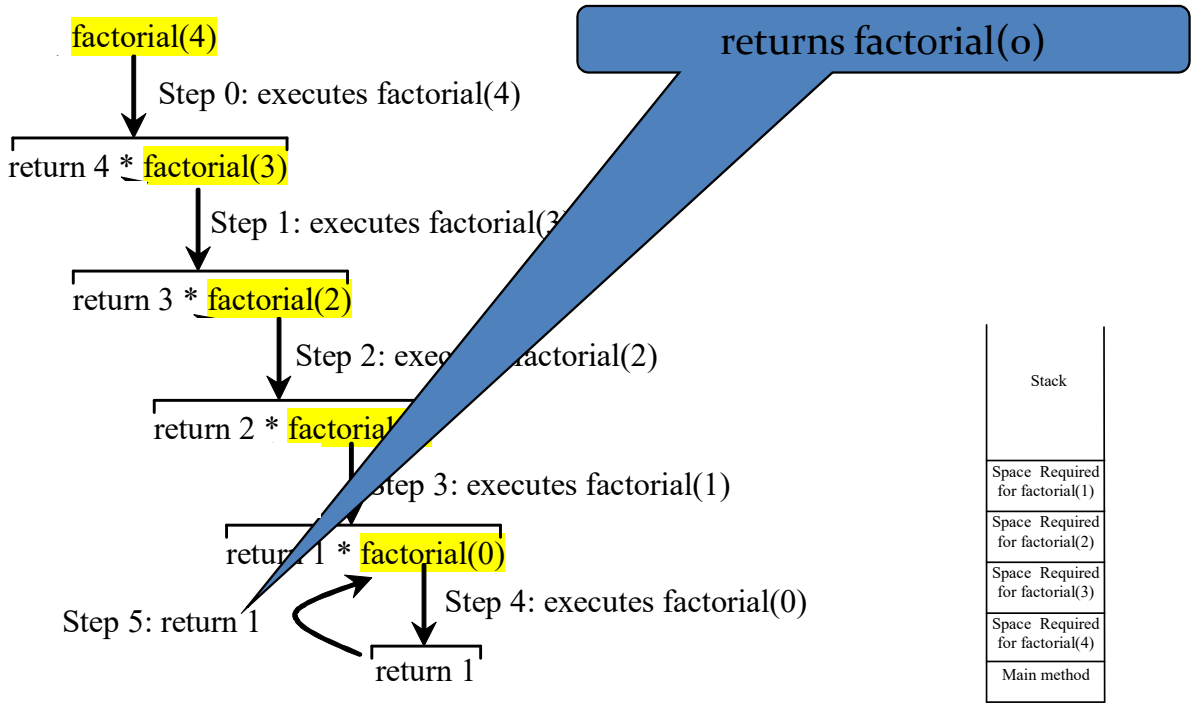
Recursive Factorial



Recursive Factorial

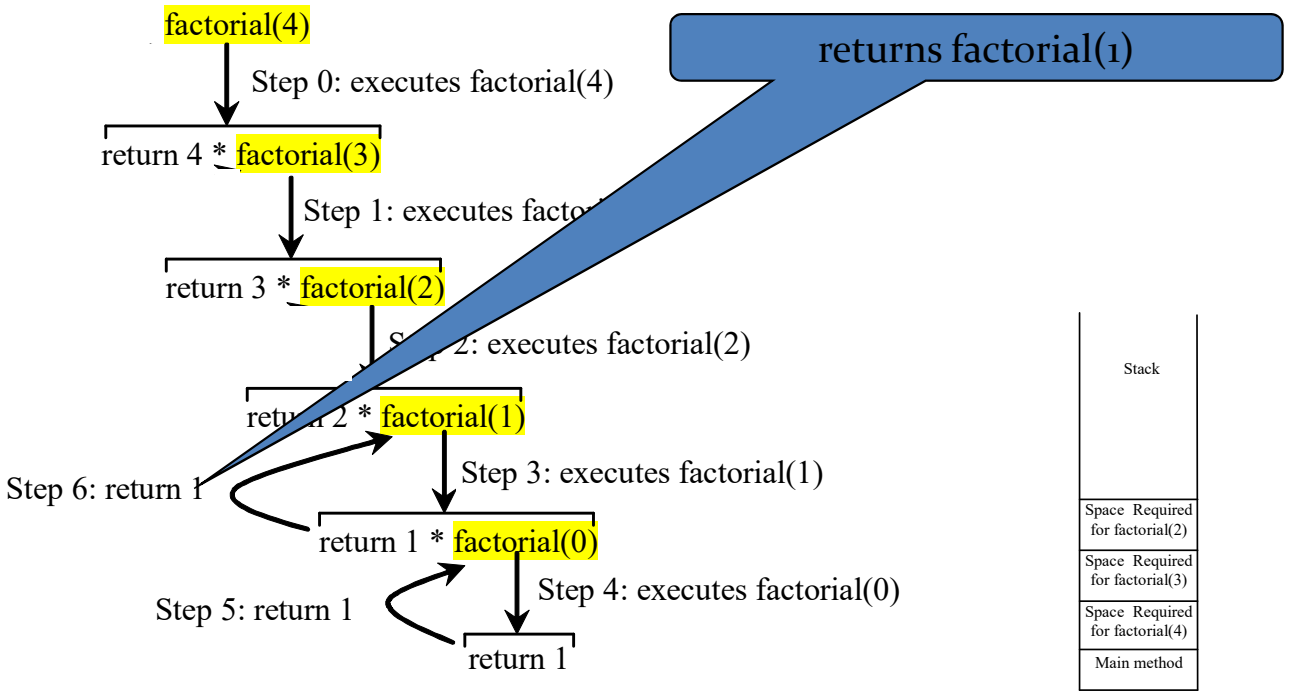


Recursive Factorial

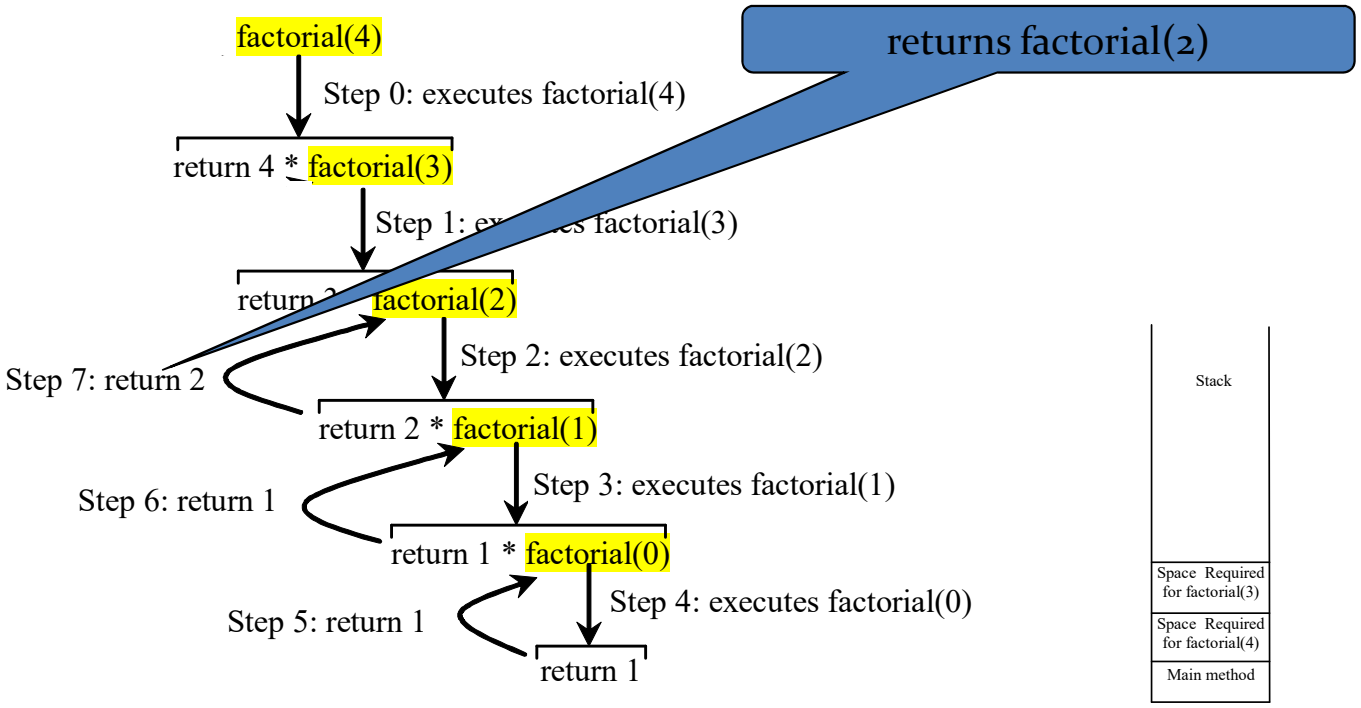


Recursive Factorial

Oregon State University
College of Engineering

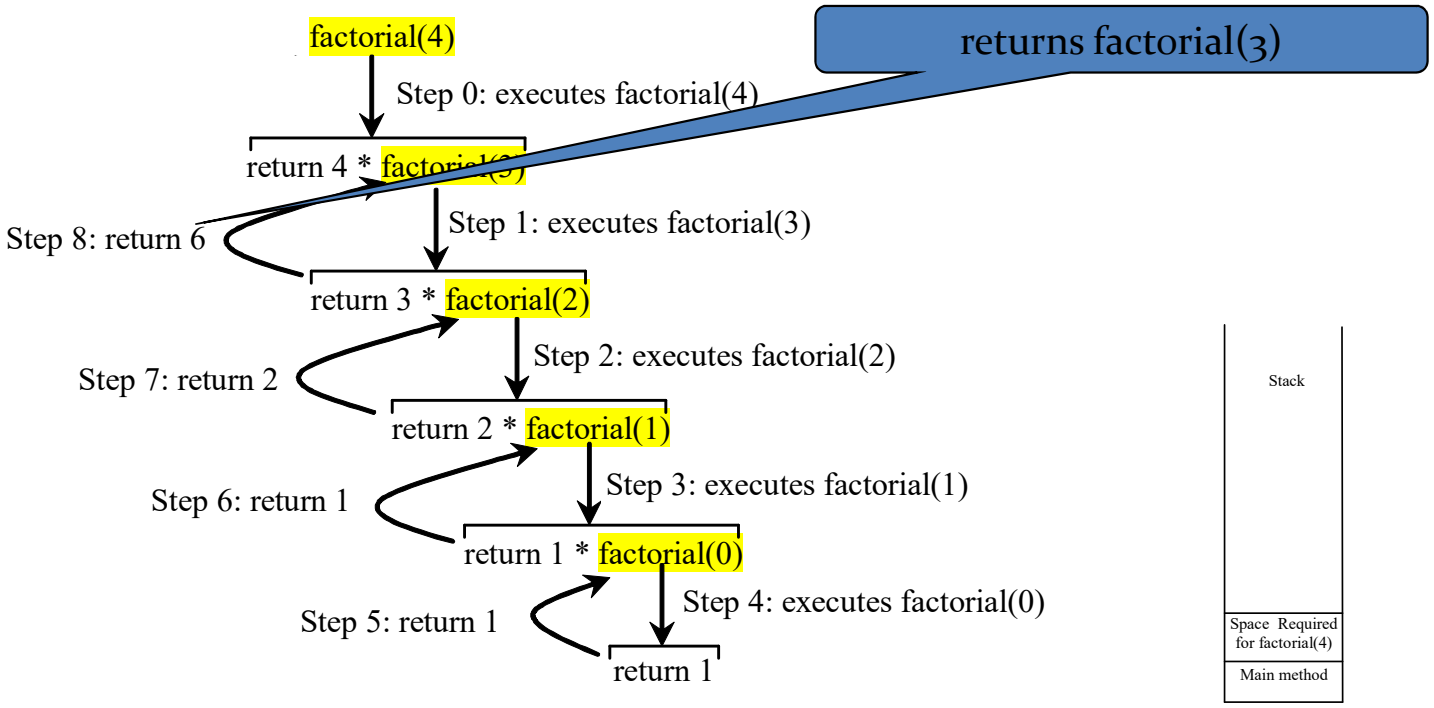


Recursive Factorial



Recursive Factorial

Oregon State University
College of Engineering



Recursive Factorial

Oregon State University
College of Engineering