CS 161
Intro to CS I

Finish Recursion/Begin Memory Model

Odds and Ends

Oregon State University
College of Engineering

- Assignment 5 posted
- Assignment 4 demo this week

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);
```

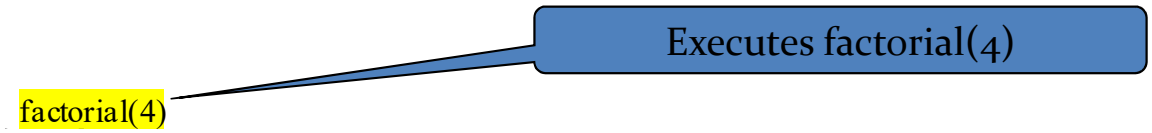
```
factorial(4) = 4 * factorial(3)
             = 4 * (3 * factorial(2))
             = 4 * ( 3 * (2 * factorial(1)))
             = 4 * ( 3 * ( 2 * (1 * factorial(0))))
             = 4 * ( 3 * ( 2 * (1 *1)))
             = 4 * ( 3 * ( 2 * 1))
             = 4 * (3 * 2)
             = 4 * 6
      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

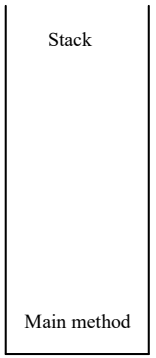
Differences

Oregon State University
College of Engineering

- Pros
  - Readability
- Cons
  - Efficiency
  - Memory

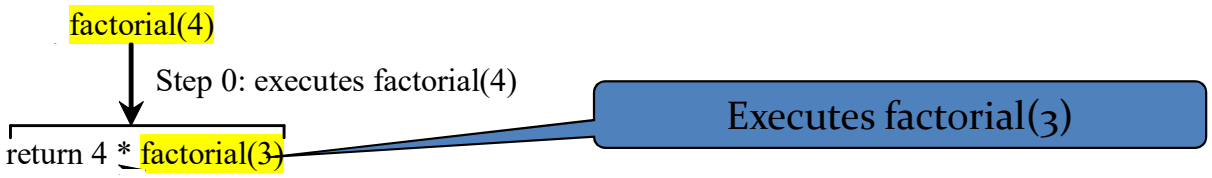
Recursive Factorial

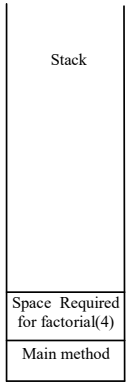




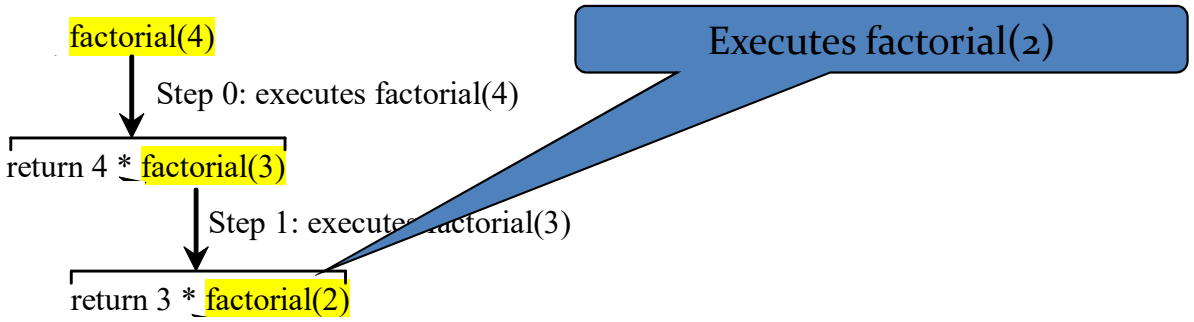
Recursive Factorial

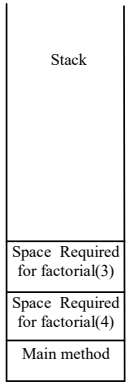
Oregon State University
College of Engineering





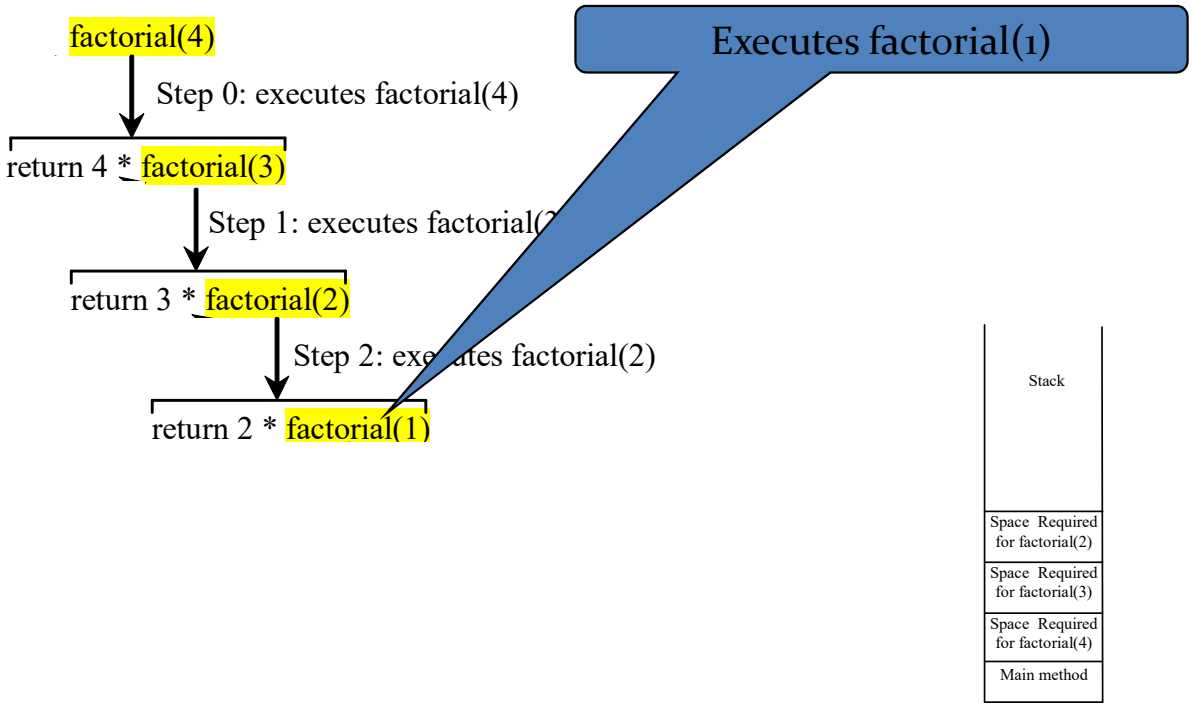
Recursive Factorial



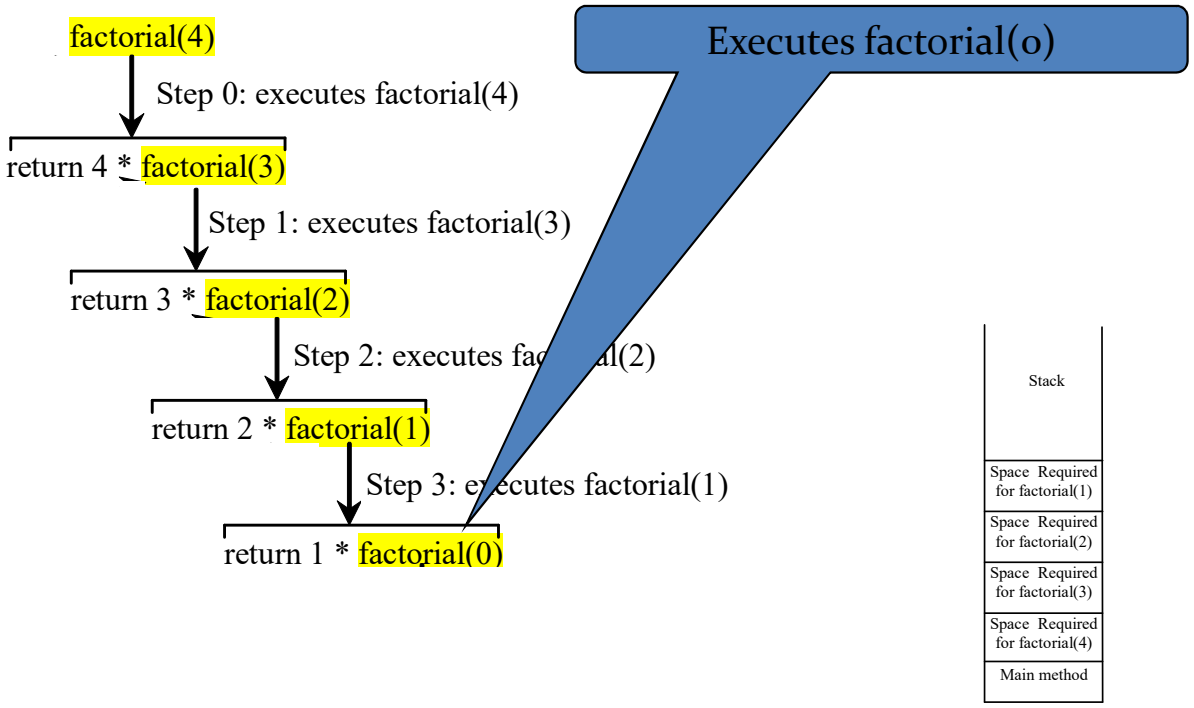


Recursive Factorial

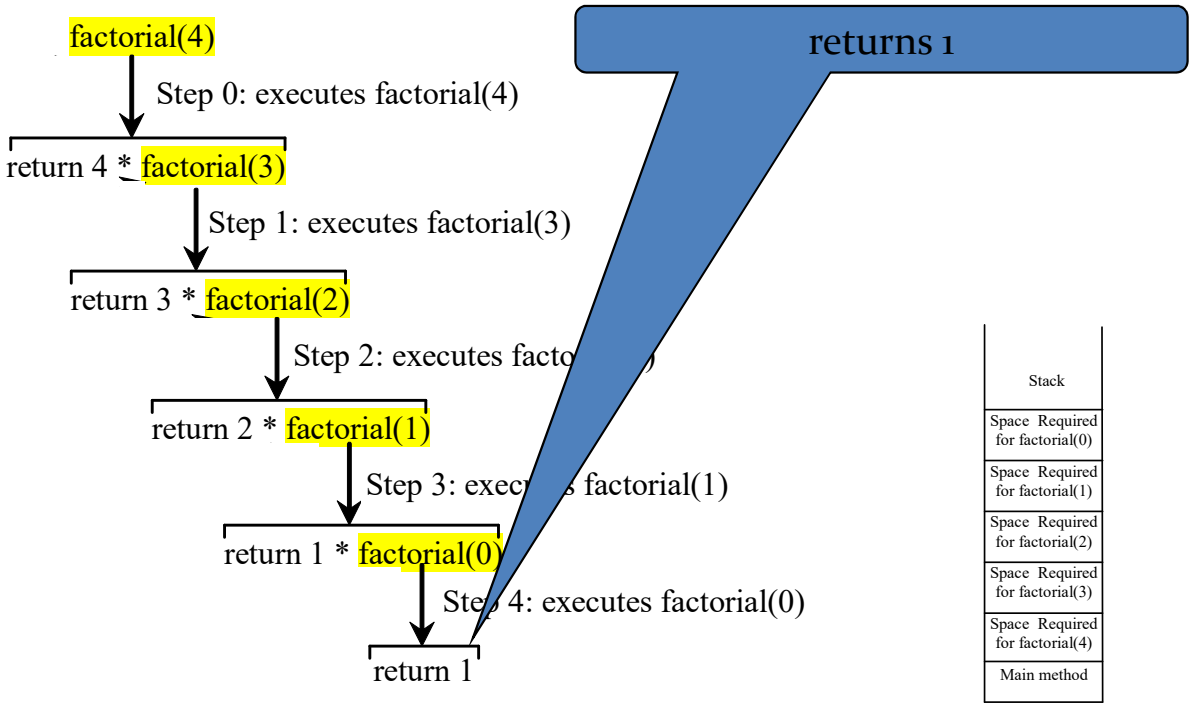
Oregon State University
College of Engineering



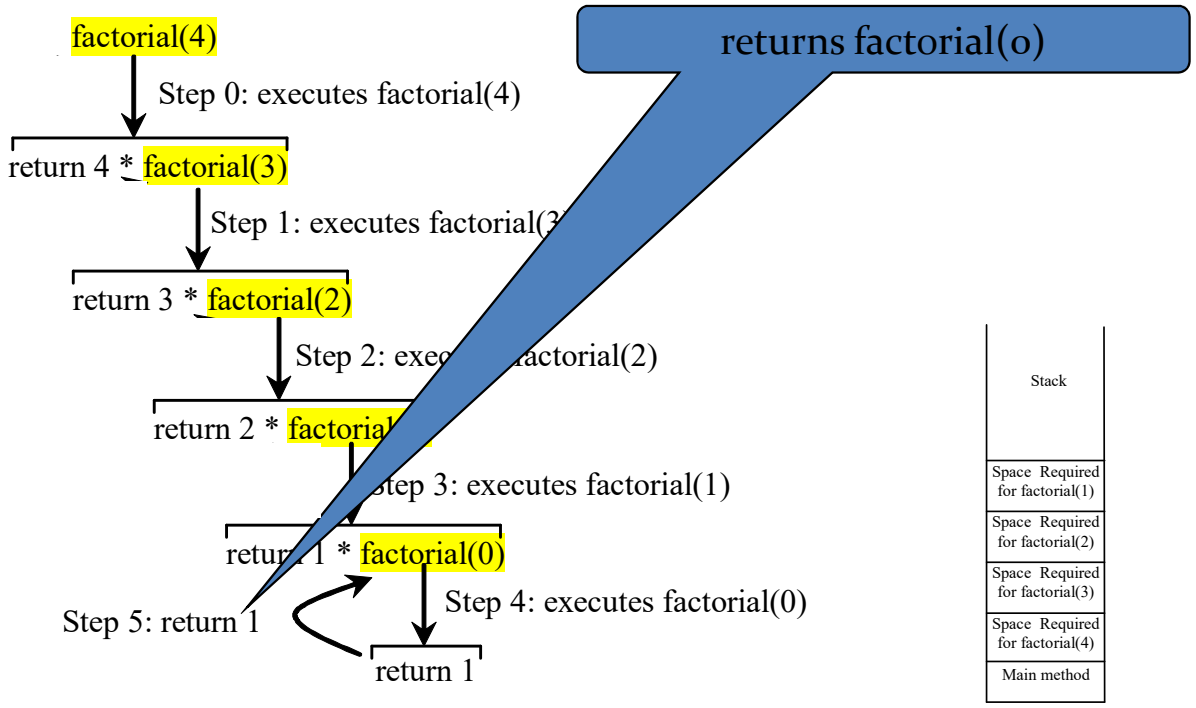
Recursive Factorial



Recursive Factorial

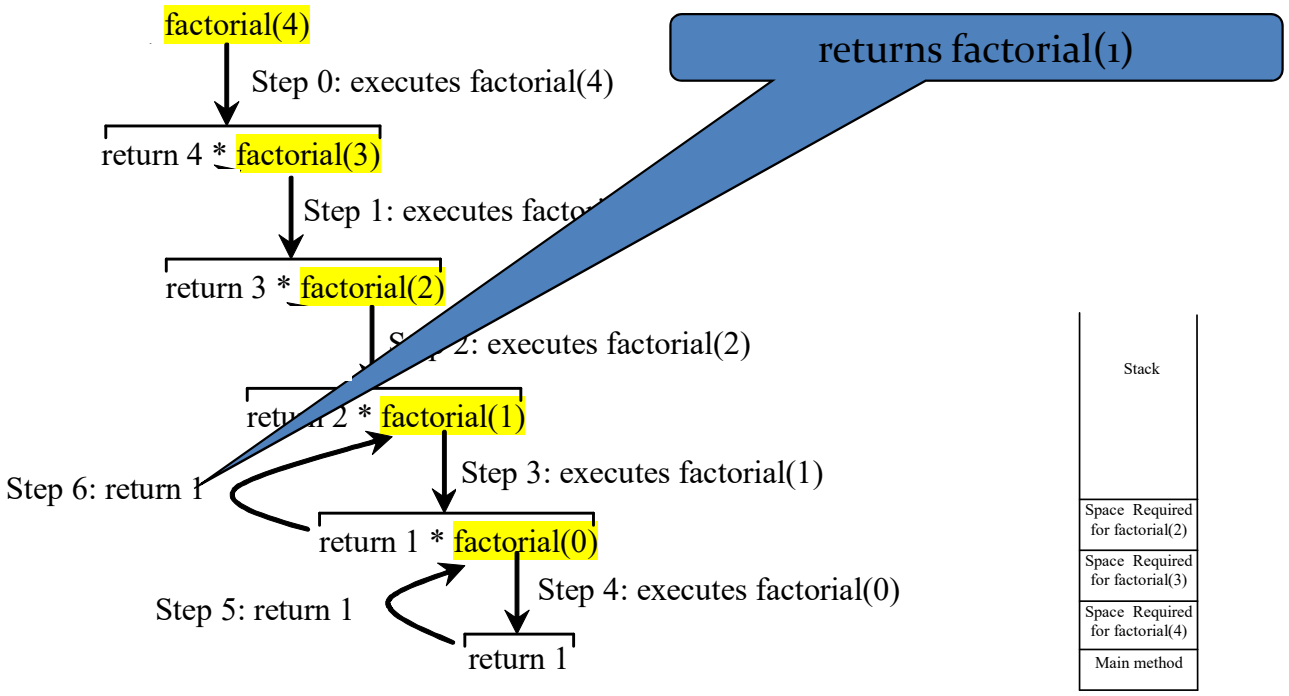


Recursive Factorial

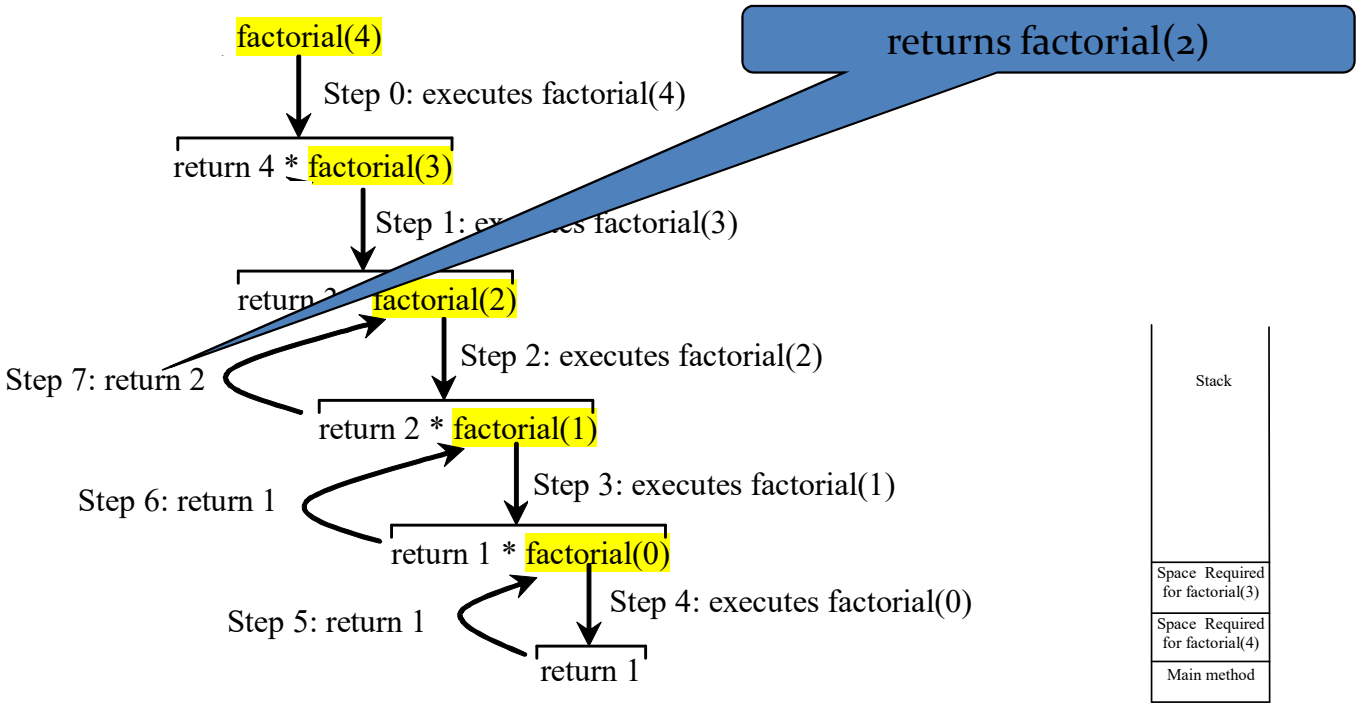


Recursive Factorial

Oregon State University
College of Engineering

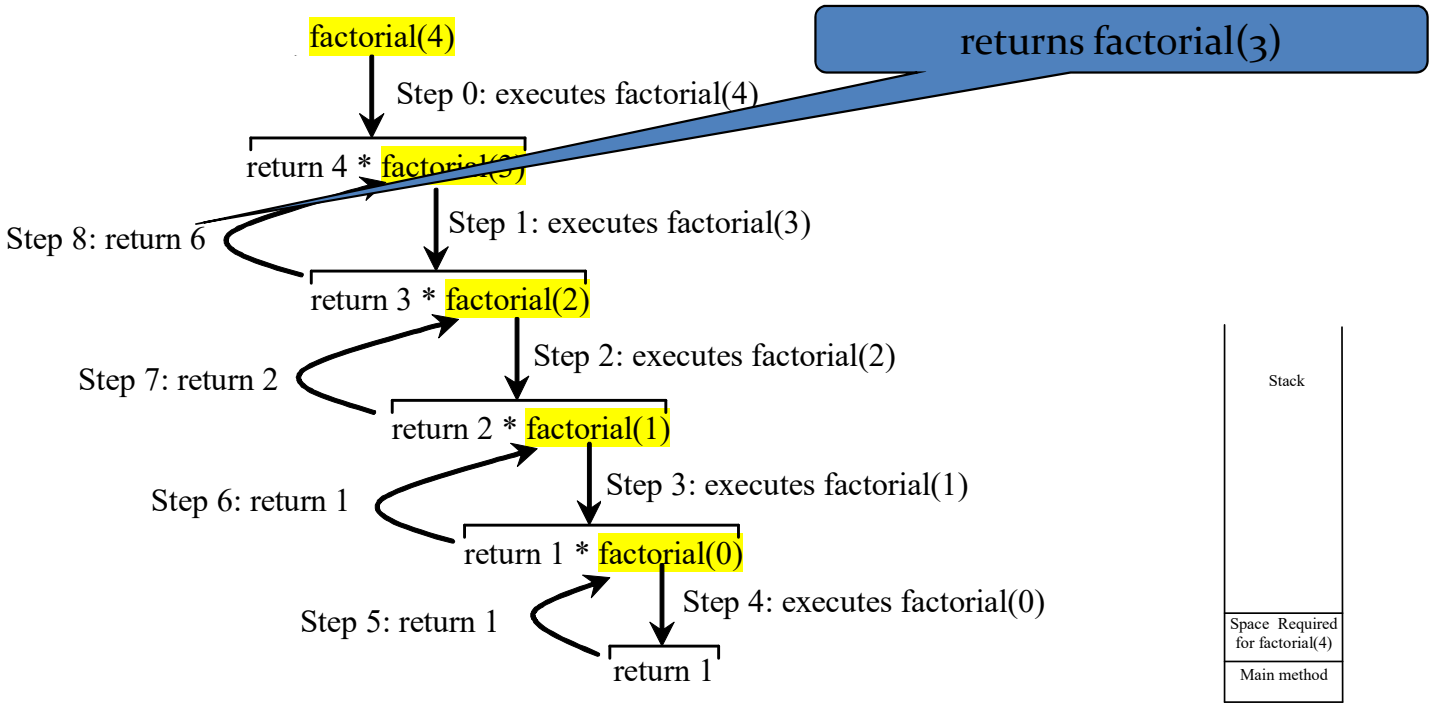


Recursive Factorial



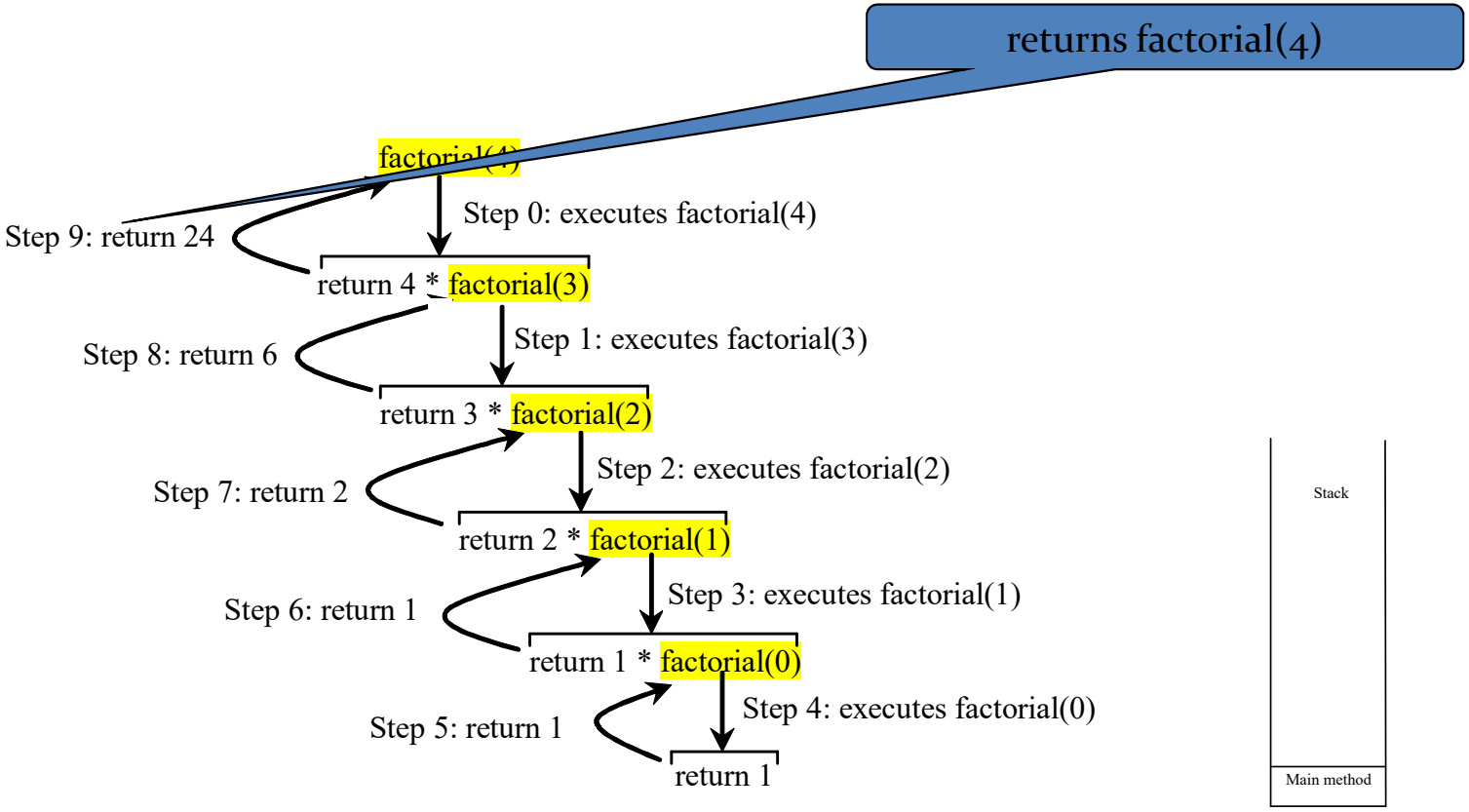
Recursive Factorial

Oregon State University
College of Engineering



Recursive Factorial

Oregon State University
College of Engineering



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…

Stack vs. Heap

Oregon State University
College of Engineering

- Static vs. Dynamic

Static vs. Dynamic

- Static Semantics
  - Assign address of variable

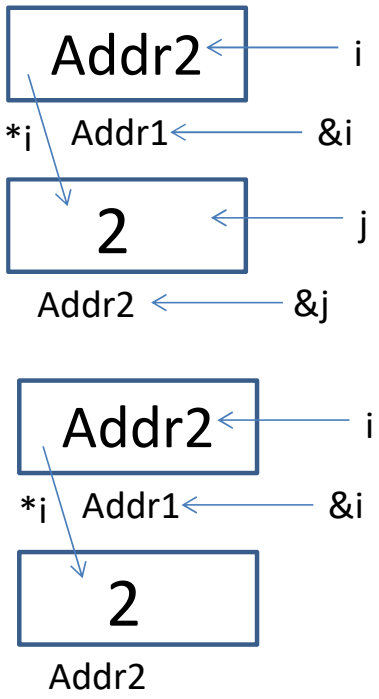
  int *i, j=2;

  i=&j;
- Dynamic Semantics
  - Create memory
  - Assign memory to pointer

  int *i=NULL;

  i=new int;

  *i=2;



What About Memory Leaks?

Oregon State University
College of Engineering

- What happens here…

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

Fixing Memory Leaks…

Oregon State University
College of Engineering

- 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
  }
}
```

Dynamic Memory Demo...

Oregon State University
College of Engineering