Lab 7: Arrays, Functions, Recursion

Agenda

- **Lecture Topic Review**
- Practice Questions
- Lab 7 assignment
- Weekly Reminders
- Q&A

Arrays

- Index arrays starting at 0
- Arrays are automatically passed to functions by reference

Character arrays

- C-strings are arrays of characters
- Compiler adds null terminator `'\0'` to indicate the end of array
- Examples of how to create C-strings
  - `char word[] = "recursion";`
  - → `{'r','e','c','u','r','s','i','o','n','\0'}`

  - `char word_2[10] = "engr151";`
  - → `{'e','n','g','r','1','5','1','\0',?,?}`

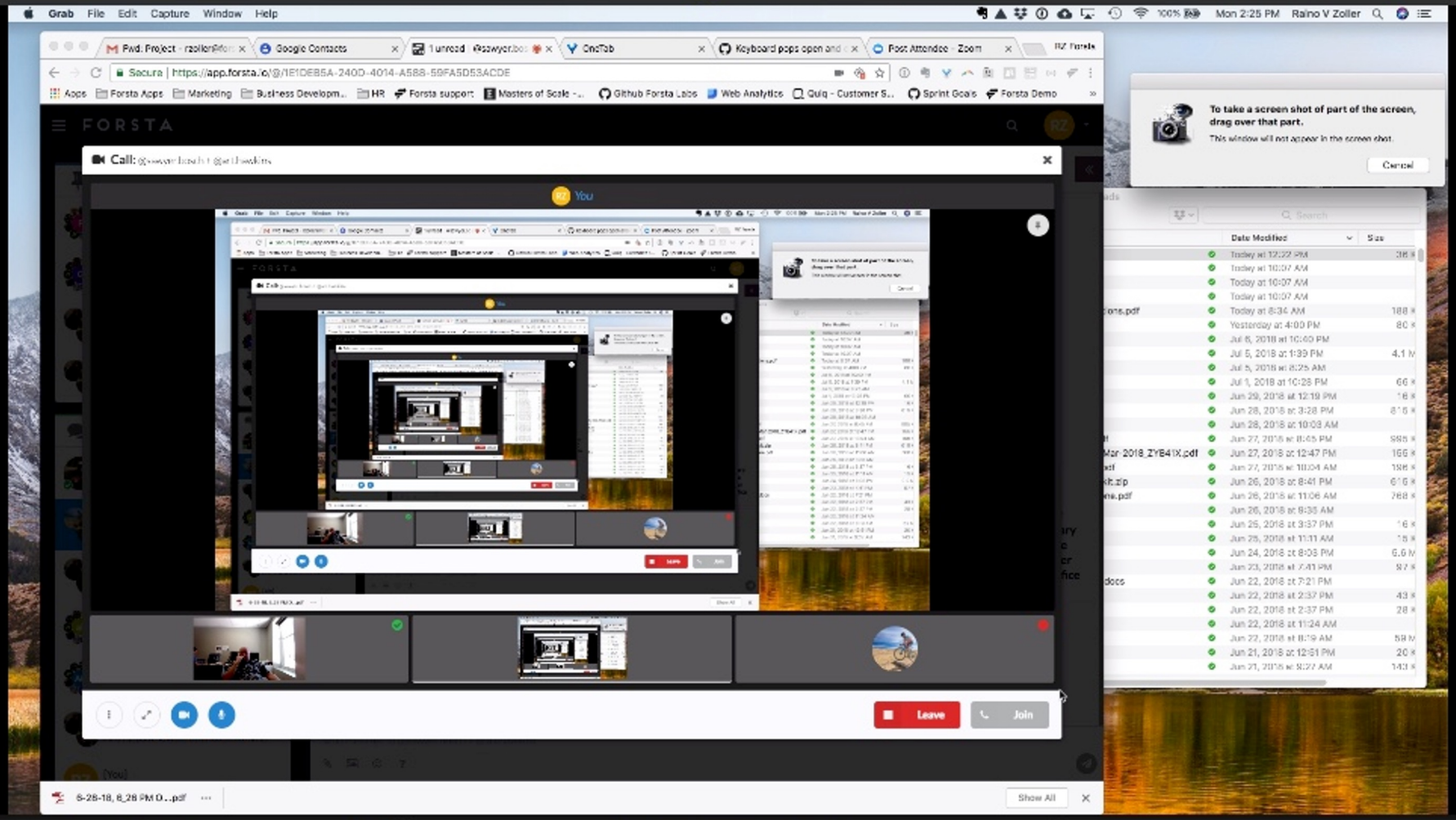
Recursion

- A program that calls itself is recursive
- Includes a general (recursive) case and a **base case**

FORSTA

Call: @sawyer.bosch + @art.hawkins

RZ You



Agenda

- Lecture Topic Review
- **Practice Questions**
- Lab 7 assignment
- Weekly Reminders
- Q&A

Problem #1: revisiting arrays

What's the output

```
int main(void) {
    int arr1[5] = {1, 2, 3, 4, 5};
    int arr2[5] = {5, 4, 3, 2, 1};

    if (arr1[3] == arr2[3]) {
    cout << "equal";
    } else if (arr1[3] > arr2[3]) {
    cout << "greater";
    } else {
    cout << "lesser";
    }
return 0;
}
```

A. greater
B. equal
C. lesser
D. nothing

Problem #1: revisiting arrays

What's the output

```
int main(void) {
    int arr1[5] = {1, 2, 3, 4, 5};
    int arr2[5] = {5, 4, 3, 2, 1};

    if (arr1[3] == arr2[3]) {
    cout << "equal";
    } else if (arr1[3] > arr2[3]) {
    cout << "greater";
    } else {
    cout << "lesser";
    }
return 0;
}
```

A. greater
B. equal
C. lesser
D. nothing

Problem #2: revisiting arrays

What's the output

```
#include <iostream>

using namespace std;

const int SIZE = 5;

void mult(int a[], int size) {
   for (int i = 0; i < size; i++) {
       a[i] = a[i] *2;
   }
}
int main(void) {
   int a[SIZE] = {1, 3, 8, 9, 10};
   cout << a[1] << " ";
   mult(a, SIZE);
   cout << a[1];
   return 0;
}
```

A. 1 2
B. 1 1
C. 3 3
D. 3 6

Problem #2

What's the output

```
#include <iostream>

using namespace std;

const int SIZE = 5;

void mult(int a[], int size) {
   for (int i = 0; i < size; i++) {
       a[i] = a[i] *2;
   }
}
int main(void) {
   int a[SIZE] = {1, 3, 8, 9, 10};
   cout << a[1] << " ";
   mult(a, SIZE);
   cout << a[1];
   return 0;
}
```

A. 1 2
B. 1 1
C. 3 3
D. 3 6

Problem #3: recursion

What's the output of the following?

```
#include <iostream>
using namespace std;
int factorial (int n) {
        if (n==0) {
                return 1;
          }
        else {
                return n * factorial(n-1);
          }
    }
int main(void) {
    cout << factorial(4) << endl;
    return 0;
}
```

A. 5
B. 120
C. 24

Problem #3: recursion solution

What's the output of the following?

```
#include <iostream>
using namespace std;
int factorial (int n) {
        if (n==0) {
                return 1;
          }
        else {
                return n * factorial(n-1);
          }
    }
int main(void) {
    cout << factorial(4) << endl;
    return 0;
}
```

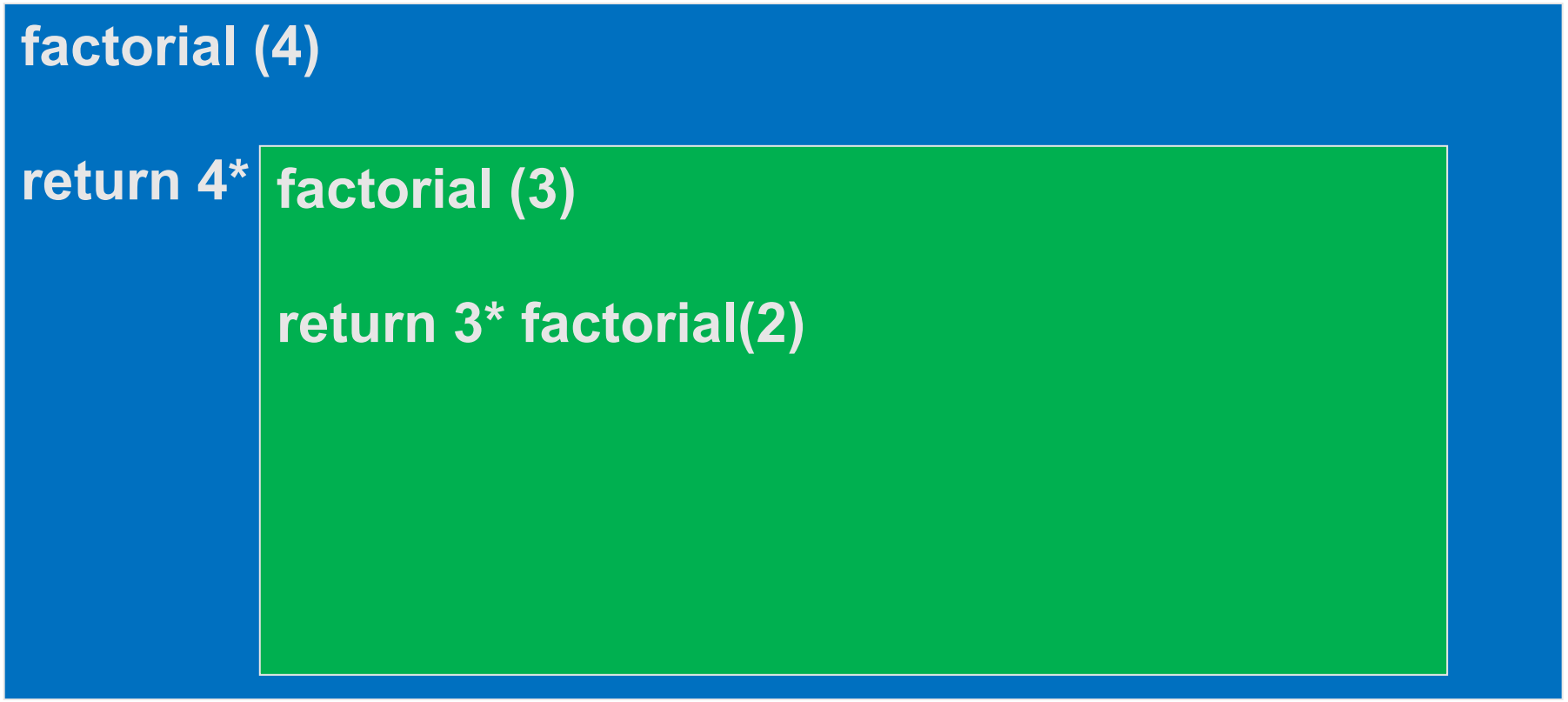
A. 5
B. 120
C. 24

Problem #3: Recursion Solution

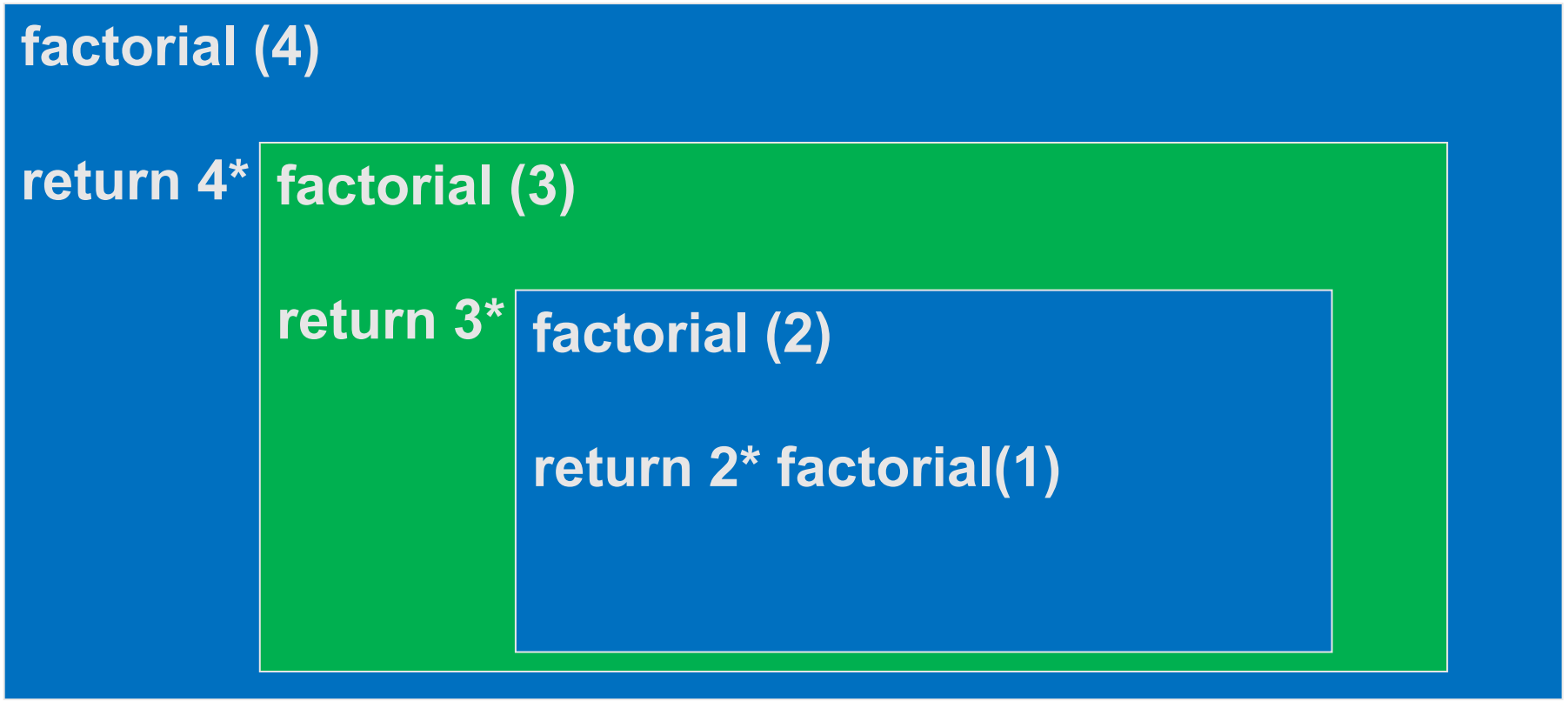
factorial (4)

return 4* factorial(3)

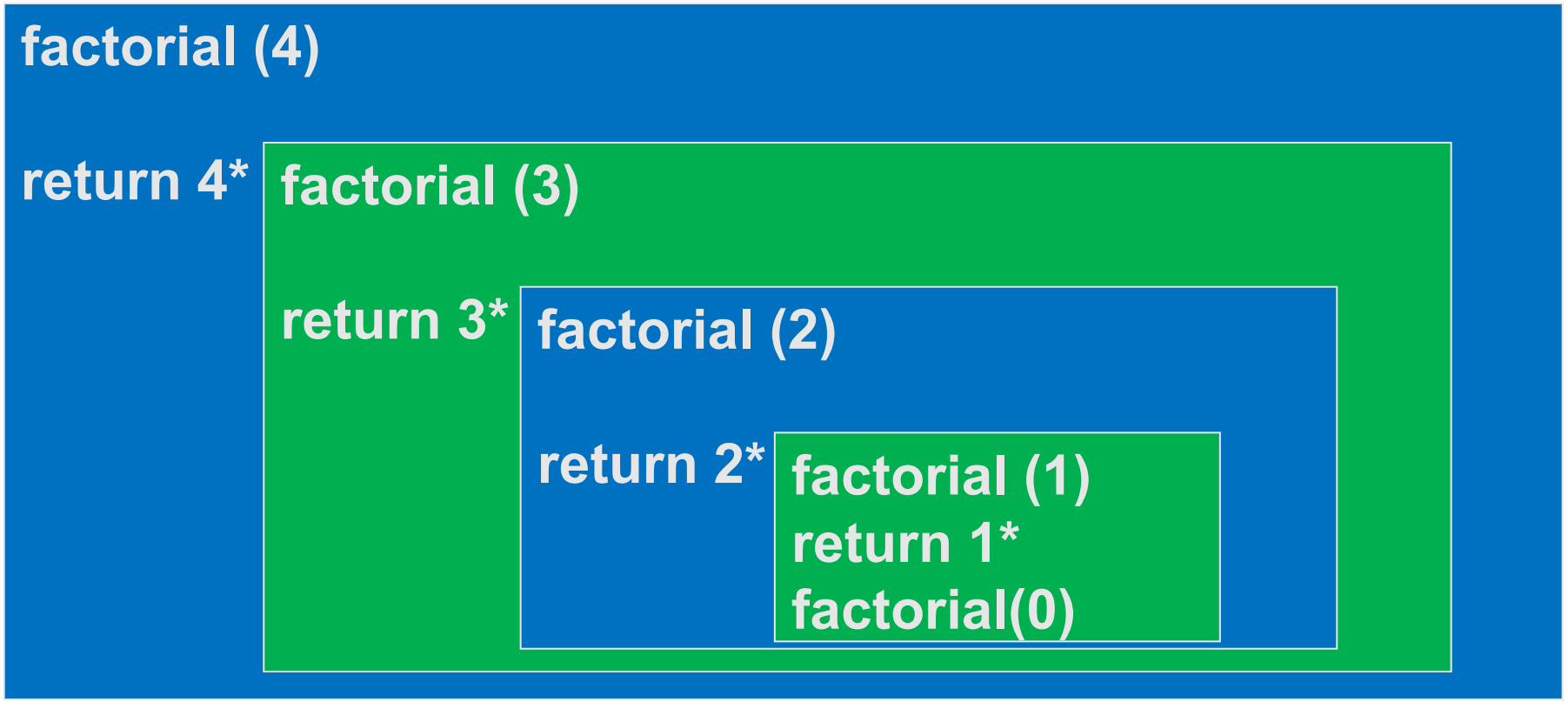
Problem #3: Recursion Solution



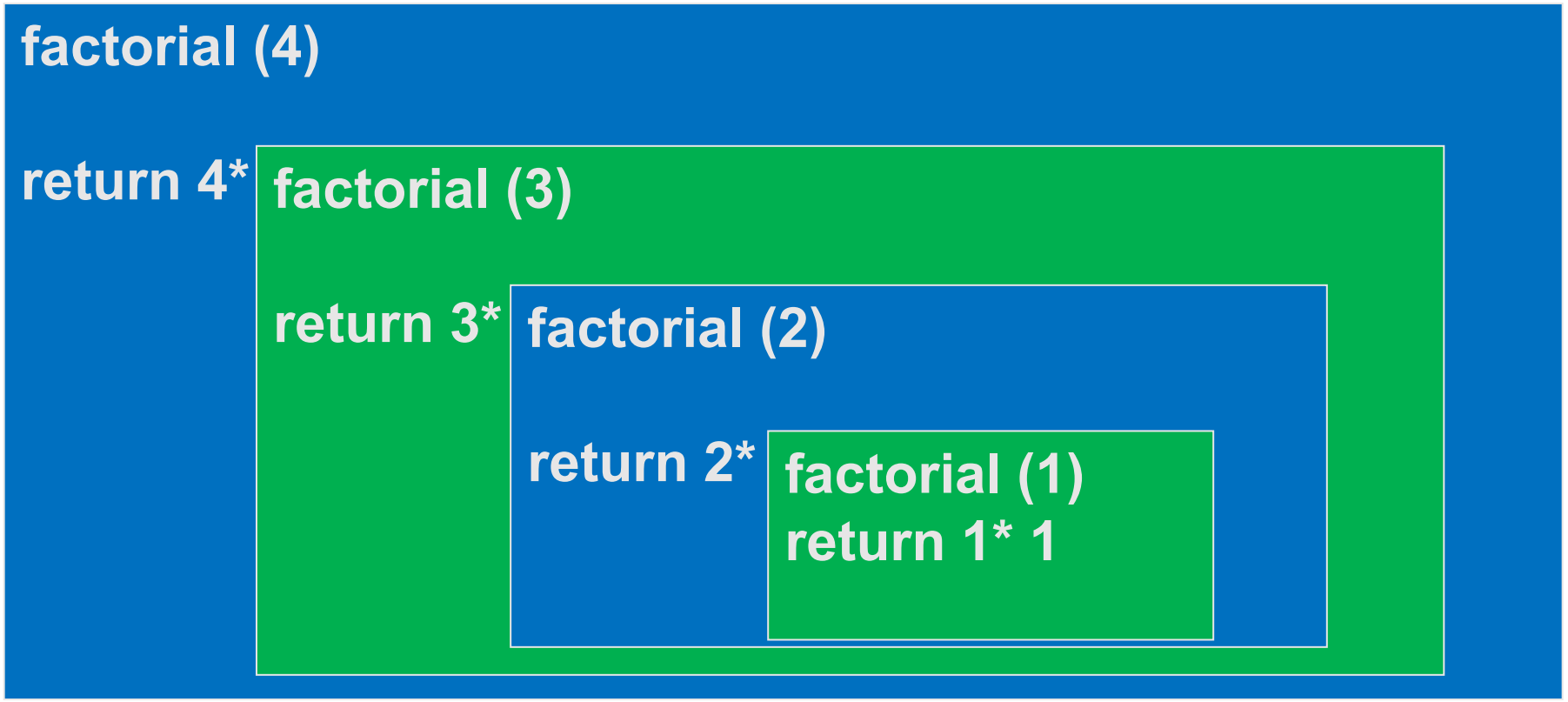
Problem #3: Recursion Solution



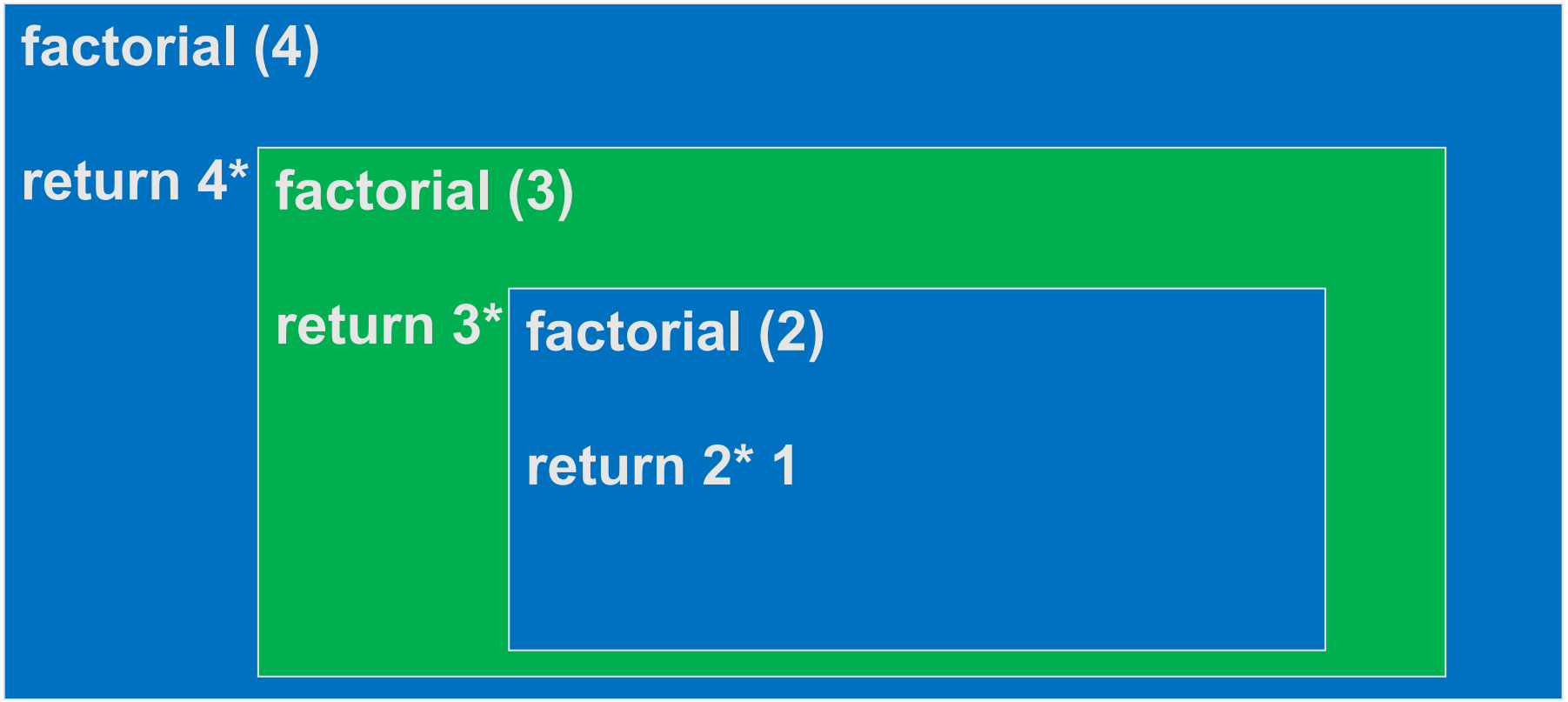
Problem #3: Recursion Solution



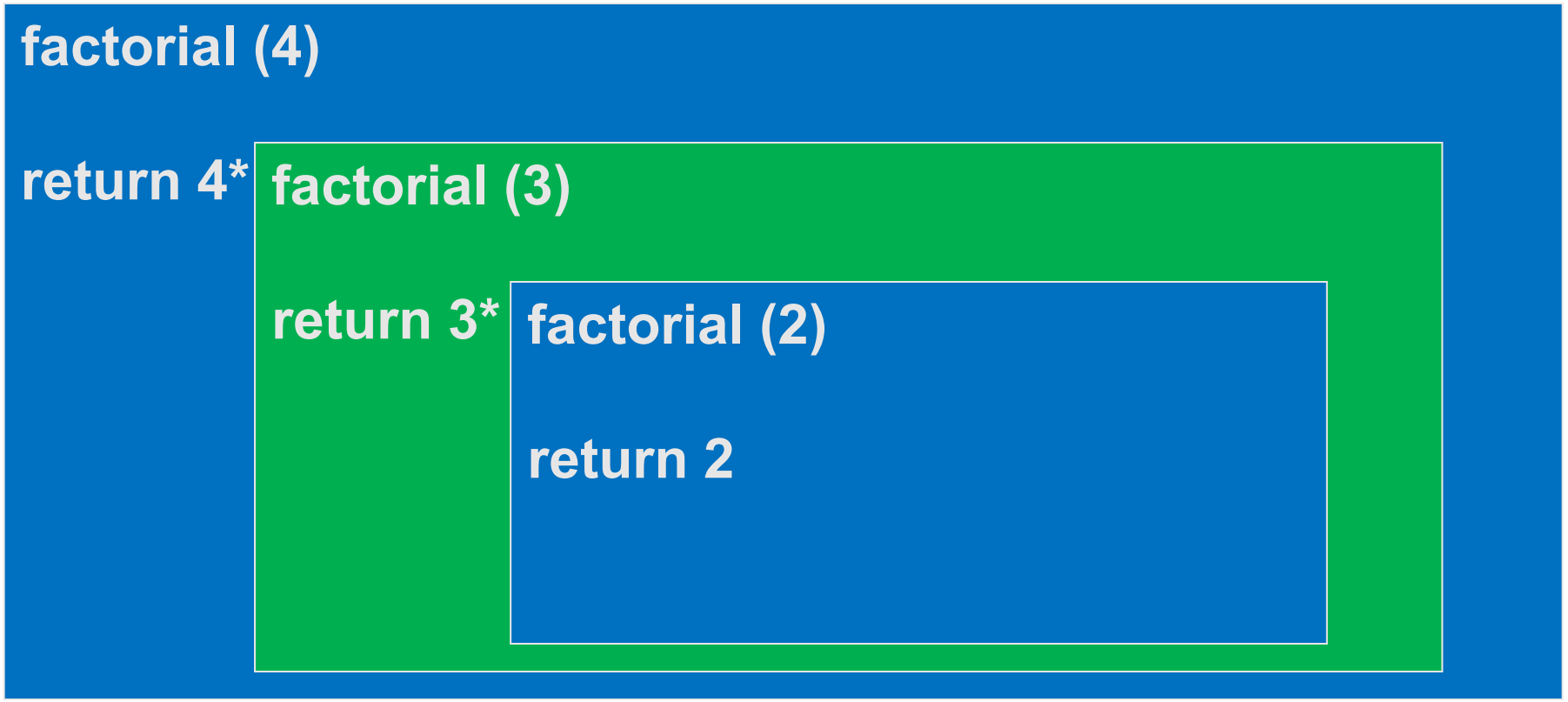
Problem #3: Recursion Solution



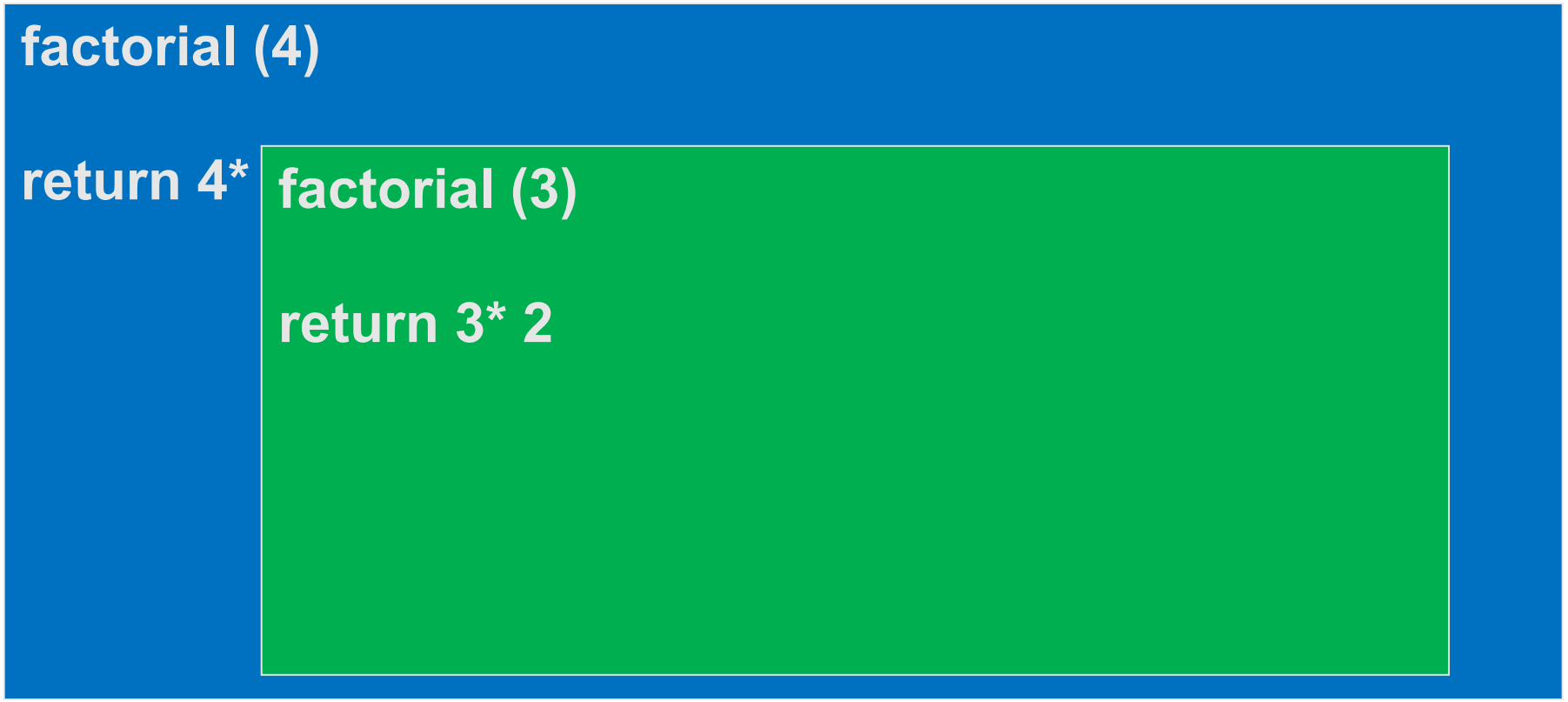
Problem #3: Recursion Solution



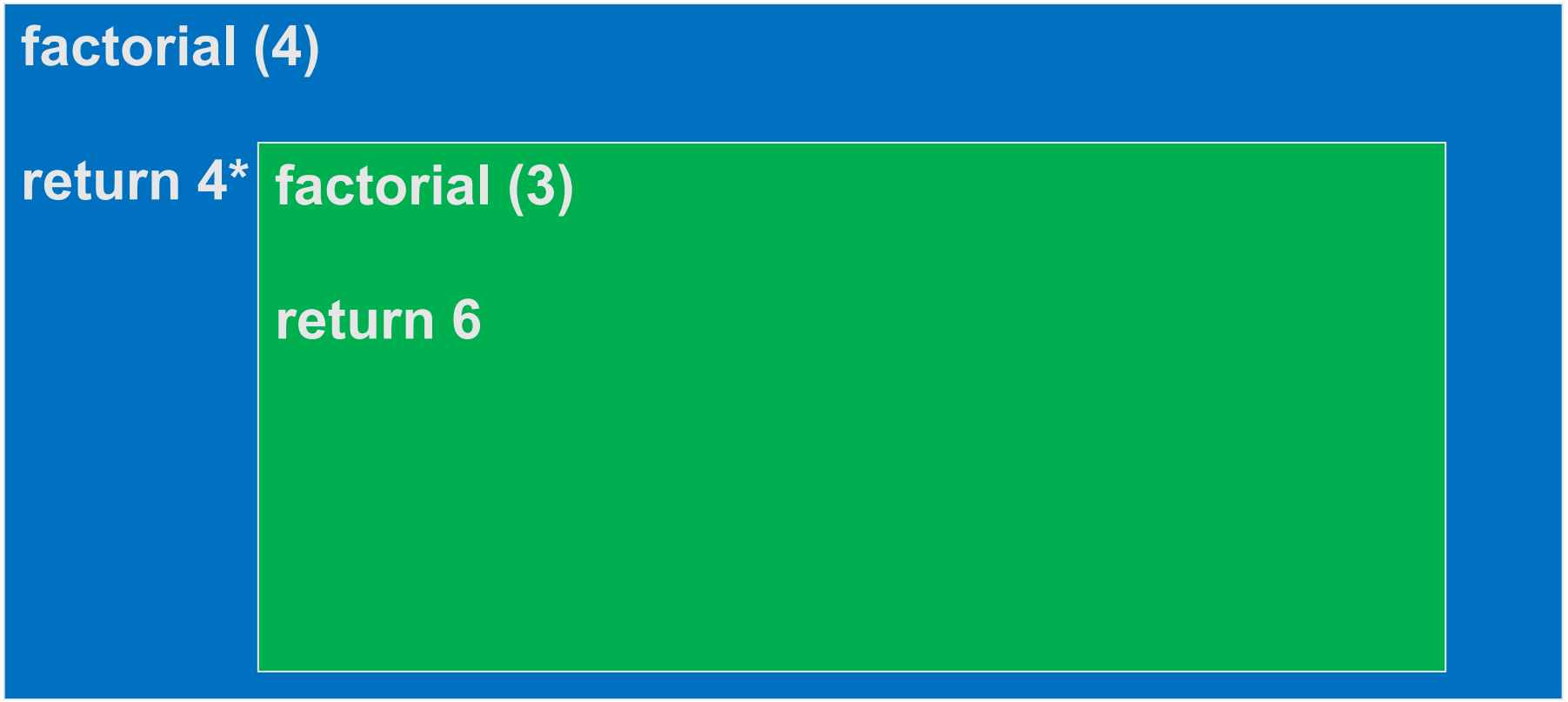
Problem #3: Recursion Solution



Problem #3: Recursion Solution



Problem #3: Recursion Solution



Problem #3: Recursion Solution

factorial (4)

return 4* 6

Problem #3: Recursion Solution

24

Problem #4: recursion

```
int matryoshkaDoll (int n) {
    cout <<  "Taking out doll # " << n << endl;
    if (n==0) {
        return n;
    }
    else {
        return matryoshkaDoll(n-1);
    }

}
int main(void) {
    matryoshkaDoll(10);
    return 0;
}
```

A

```
int matryoshkaDoll (int n) {
    if (n==0) {
        return n;
    }
    else {
        return matryoshkaDoll(n-1);
    }

}
int main(void) {
    cout << "Taking out doll #  " << matryoshkaDoll(10)<< endl;
    return 0;
}
```

B

```
int matryoshkaDoll (int n) {
    if (n==0) {
        return n;
    }
    else {
        cout <<  "Taking out doll # " << n << endl;
        return matryoshkaDoll(n-1);
    }

}
int main(void) {
    matryoshkaDoll(10);
    return 0;
}
```

C

Matryoshka Dolls (Russian dolls) are a well-known toy. Say we want to write a recursion function that prints out the following output:

```
Taking out doll # 10
Taking out doll # 9
Taking out doll # 8
Taking out doll # 7
Taking out doll # 6
Taking out doll # 5
Taking out doll # 4
Taking out doll # 3
Taking out doll # 2
Taking out doll # 1
Taking out doll # 0
```

Which option is the right implementation?

Problem #4: recursion

```
int matryoshkaDoll (int n) {
    cout <<  "Taking out doll # " << n << endl;
    if (n==0) {
        return n;
    }
    else {
        return matryoshkaDoll(n-1);
    }

}
int main(void) {
    matryoshkaDoll(10);
    return 0;
}
```

A

```
int matryoshkaDoll (int n) {
    if (n==0) {
        return n;
    }
    else {
        return matryoshkaDoll(n-1);
    }

}
int main(void) {
    cout << "Taking out doll #  " << matryoshkaDoll(10)<< endl;
    return 0;
}
```

B

```
int matryoshkaDoll (int n) {
    if (n==0) {
        return n;
    }
    else {
        cout <<  "Taking out doll # " << n << endl;
        return matryoshkaDoll(n-1);
    }

}
int main(void) {
    matryoshkaDoll(10);
    return 0;
}
```

C

Matryoshka Dolls (Russian dolls) are a well-known toy. Say we want to write a recursion function that prints out the following output:

```
Taking out doll # 10
Taking out doll # 9
Taking out doll # 8
Taking out doll # 7
Taking out doll # 6
Taking out doll # 5
Taking out doll # 4
Taking out doll # 3
Taking out doll # 2
Taking out doll # 1
Taking out doll # 0
```

Which option is the right implementation?

Agenda

- Lecture Topic Review
- Practice Questions
- **Lab 7 assignment**
- Weekly Reminders
- Q&A

Today’s lab

- Part 1: printing a 2D array
  - Tip: Iterate through 2D array along each row. For a fixed row, iterate along each column.
- Part 2: working with char arrays
  - Task 1: Find length of character array.
    - Tip: What are all character arrays supposed to end with?
  - Task 2: Concatenate two words.
  - Task 3: Check upper and lower case.
    - Tip: What is the the result of ‘a’ - ‘A’ and ‘k’ - ‘K’? Is this difference common across all letters?
- Part 3: recursive Fibonacci sequence
  - Tip: Identify base cases similar to fib_for and fib_while.
  - Tip: Revisit the example of factorial computation through recursion.

Today's lab

Revisiting Test Driven Development

- One of the exercises is writing a function that concatenates two c-strings, storing the concatenated result in "word1":

```
void concatenate(char word1[], char word2[])
```

- Group brainstorm:

  What do we need to require of the argument array sizes?

  What are some potential pitfalls?

Today's lab

Revisiting Test Driven Development

- One of the exercises is writing a function that concatenates two c-strings, storing the concatenated result in "word1":

```
void concatenate(char word1[], char word2[])
```

- Group brainstorm:

  What do we need to require of the argument array sizes? The first argument must be of size word1 + word2 + 1.
  What are some potential pitfalls? Missing null terminator, wrong starting index for concatenation

Agenda

- Lecture Topic Review
- Practice Questions
- Lab 7 assignment
- **Weekly Reminders**
- **Q&A**

Weekly Reminders

- Lab 7 is due before Lab next week.
- Graded Midterms have been released
- Regrade requests for the Midterm can be submitted no later than Friday, October 27th 11:59pm

Questions?