

# 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";`
    - $\rightarrow \{ 'r', 'e', 'c', 'u', 'r', 's', 'i', 'o', 'n', \underline{'\0'} \}$
  - `char word_2[10] = "engr151";`
    - $\rightarrow \{ 'e', 'n', 'g', 'r', '1', '5', '1', '\0', \underline{?}, \underline{?} \}$

# Recursion

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

Call: @sawyer.bosch + @art.hawkins



Grab File Edit Capture Window Help

RZ You

Secura | <https://app.forsta.io/@/1E1DEB5A-240D-4014-A588-55FA5DE5ACDE>

Apps Forsta Apps Marketing Business Developm... HR Forsta support Masters of Scale ... Github Forsta Labs Web Analytics Cliq - Customer S... Sprint Goals Forsta Demo

Call: @sawyer.bosch + @art.hawkins

To take a screen shot of part of the screen, drag over that part. This window will not appear in the screen shot. Cancel

Date Modified	Size
Today at 12:22 PM	218 K
Today at 10:07 AM	
Today at 10:07 AM	
Today at 10:07 AM	
Today at 8:34 AM	188 K
Yesterday at 4:00 PM	80 K
Jul 5, 2018 at 10:40 PM	
Jul 5, 2018 at 1:39 PM	4.1 M
Jul 5, 2018 at 8:25 AM	
Jul 1, 2018 at 10:28 PM	66 K
Jun 29, 2018 at 12:19 PM	18 K
Jun 28, 2018 at 3:28 PM	870 K
Jun 28, 2018 at 10:08 AM	
Jun 27, 2018 at 8:45 PM	985 K
Jun 27, 2018 at 12:47 PM	166 K
Jun 27, 2018 at 10:04 AM	138 K
Jun 26, 2018 at 8:41 PM	676 K
Jun 26, 2018 at 11:06 AM	768 K
Jun 26, 2018 at 9:35 AM	
Jun 25, 2018 at 3:37 PM	16 K
Jun 25, 2018 at 11:11 AM	78 K
Jun 24, 2018 at 8:06 PM	6.6 M
Jun 23, 2018 at 7:41 PM	82 K
Jun 22, 2018 at 7:21 PM	
Jun 22, 2018 at 2:37 PM	43 K
Jun 22, 2018 at 2:37 PM	28 K
Jun 22, 2018 at 11:24 AM	
Jun 22, 2018 at 8:19 AM	5.9 M
Jun 21, 2018 at 12:51 PM	20 K
Jun 21, 2018 at 8:27 AM	143 K

6-28-18\_8:28 PM D...pdf Show All



# 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

**factorial (4)**

**return 4\* factorial (3)**

**return 3\* factorial(2)**

# Problem #3: Recursion Solution

**factorial (4)**

**return 4\* factorial (3)**

**return 3\* factorial (2)**

**return 2\* factorial(1)**



# Problem #3: Recursion Solution

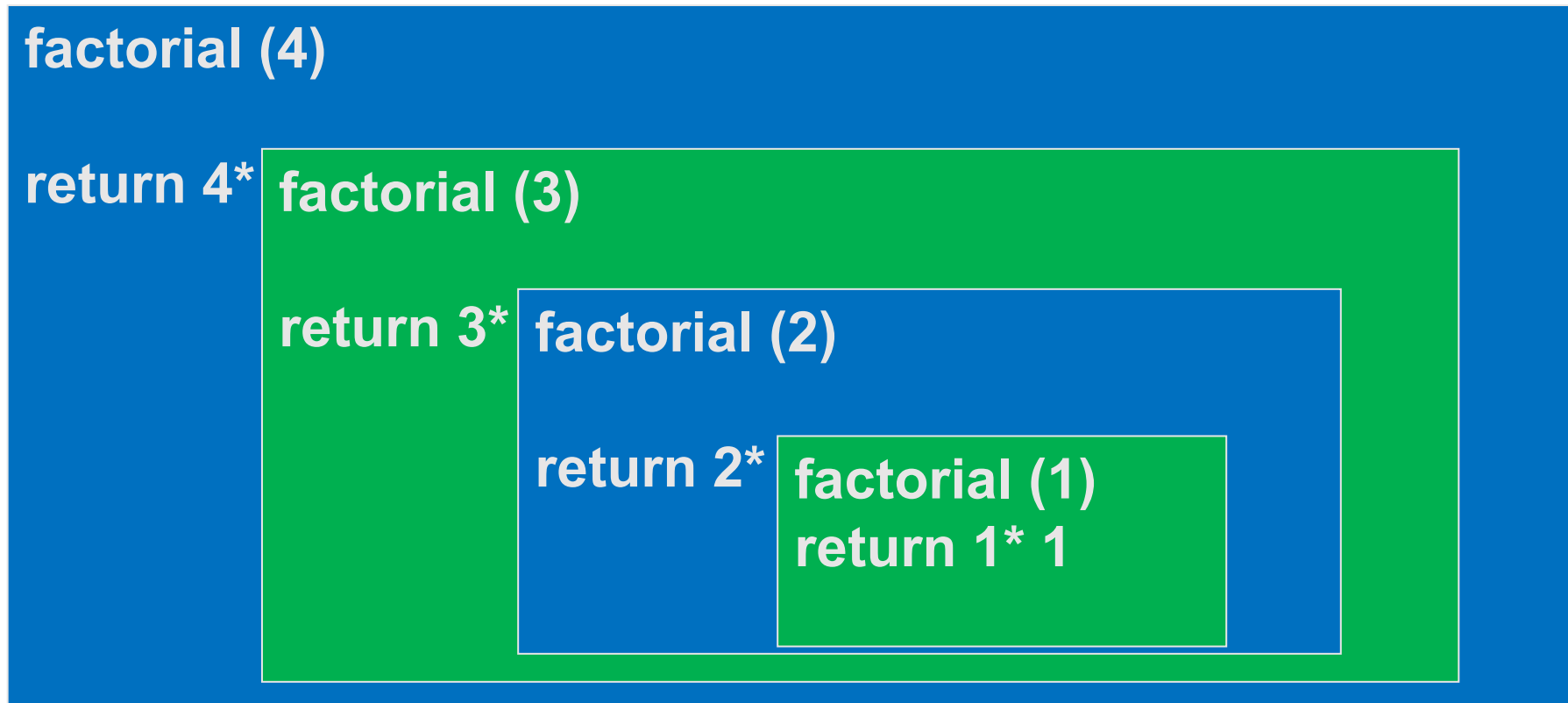
**factorial (4)**

**return 4\* factorial (3)**

**return 3\* factorial (2)**

**return 2\* factorial (1)**  
**return 1\***  
**factorial(0)**

# Problem #3: Recursion Solution



# Problem #3: Recursion Solution

**factorial (4)**

**return 4\* factorial (3)**

**return 3\* factorial (2)**

**return 2\* 1**

# Problem #3: Recursion Solution

**factorial (4)**

**return 4\* factorial (3)**

**return 3\* factorial (2)**

**return 2**

# Problem #3: Recursion Solution

**factorial (4)**

**return 4\* factorial (3)**

**return 3\* 2**

# Problem #3: Recursion Solution

**factorial (4)**

**return 4\* factorial (3)**

**return 6**

# 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;  
}  
  
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;  
}
```

B

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;  
}
```



```
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;  
}  
  
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;  
}
```



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?