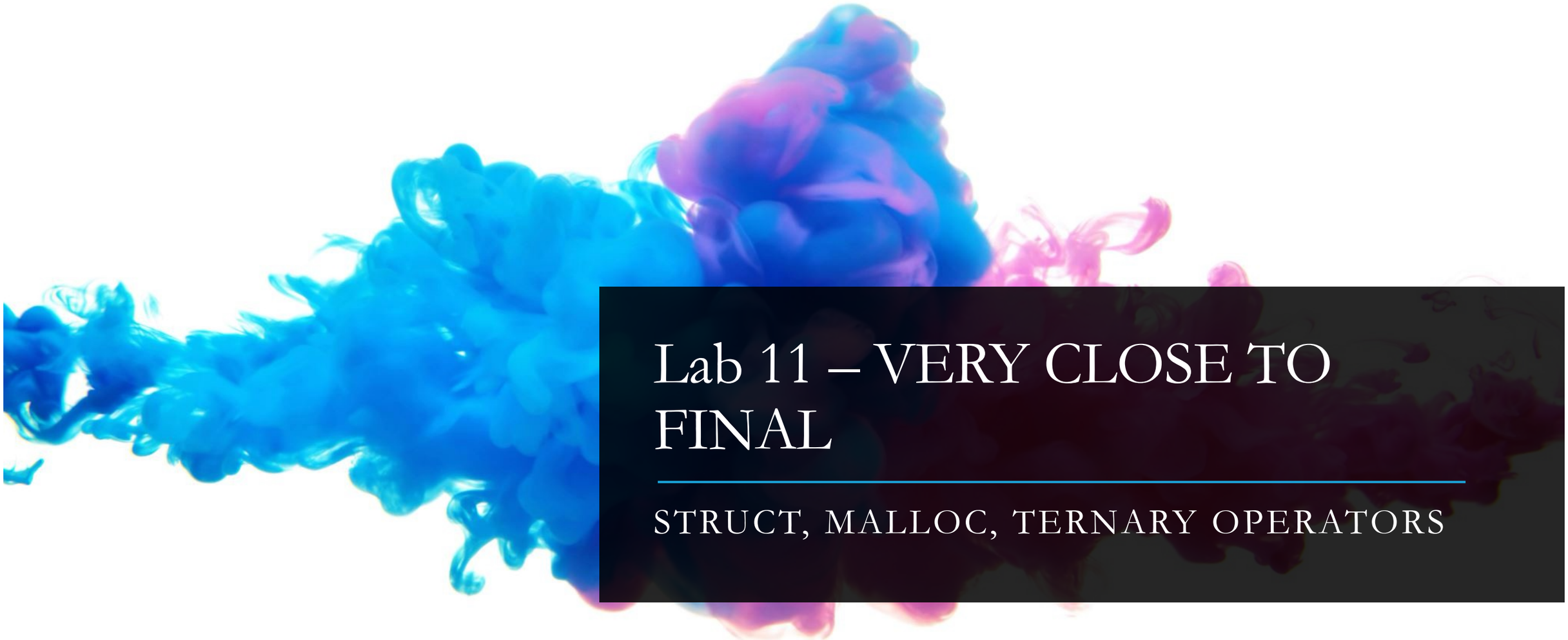


Lab 11 – VERY CLOSE TO FINAL

STRUCT, MALLOC, TERNARY OPERATORS

Correction on memset

```
     void *
     memset(void *b, int c, size_t len);

DESCRIPTION
     The memset() function writes len bytes of value c (converted to an
     unsigned char) to the string b.

RETURN VALUES
     The memset() function returns its first argument.
```

==> Assume call memset(myIntArr, 1, 5); <==

Should only be used with characters since it sets byte by byte.

- To use this with ints, it would be very complicated and is never typically used.
- So if you get a large number, that is because you are actually setting 0x1010… which is very large.

I know last time I said you could use this, but only use for zeroing an array or structure out.

- Mathematically you can; practically, you cannot not.

Struct (Good Examples)

```
typedef struct _player {
    int ID;
    int score;
    int misses[100];
    int hits[100];
    Cell board[10][10];
} Player;
```

```
typedef struct _cell {
    bool isHit;
    bool hasShip;
    CellType type; // enum
    BSCoordinate coordinate; // struct: x, y
} Cell;
```

Enum (of type unsigned char)

Unlike my code, be sure all fields are capitalized. See next page for proper styling

```
typedef enum _cellType : unsigned char {
    carrier, battleship, cruiser, submarine, destroyer,
    hit,
    miss,
    empty
} CellType;
```

Enum (Default Type int)

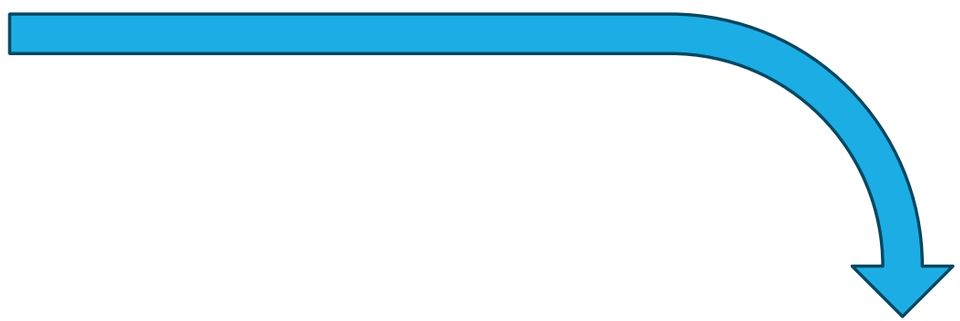
```
typedef enum _counter {
    ZERO,
    ONE,
    TWO,
} Counter;
```

Typed Enum (unsigned char)

```
typedef enum _asciiTb :
unsigned char {
    NUL, // NULL
    SOH, // Start of Heading
    STX, // Start of Text
} AsciiTb;
```

Ternary Operator (not req to know)

```
int x = 0; int y = 0;
if (x == 100) {
    y = 10;
} else {
    y = 20;
}
```



```
y = (x == 100) ? 10 : 20;
```

Malloc (Not Required)

```
#include "stdlib.h"

Employee * payrollList = (Employee*)malloc(sizeof(Emloyee) * 200); // 200 eles
if (!payrollList) {
    return -1;
}
… Use it as you would an array …

free(payrollList);
```

Commenting Functions

You will lose minimal points today, but grade will be harmed in 300+ level courses. Internships will not go well.

Be sure to include:

- What it does
- Parameter name(s) and purpose
- Pre/post condition
- What is returned
- Date updated

Commenting – Preconditions & Postconditions

- Precondition
  - What must be true **immediately** prior to executing the function?
  - Do values need to be in a certain range?
  - Does something need to be non-null?
- Postcondition
  - What must be true **immediately** after executing the function?
  - Is the value NULL or freed?
  - Did a value change?

Functions You SHOULD Use

```
char *fgets(char * __restrict, int, FILE *);
```

```
int fscanf(FILE * __restrict, const char * __restrict, ...);
```

```
int fputs(const char * __restrict, FILE * __restrict);
```

```
int fprintf(FILE * __restrict, const char * __restrict, ...);
```

```
FILE *fopen(const char * __restrict __filename, const char * __restrict __mode)
```

```
int fclose(FILE *);
```

```
int feof(FILE *);
int fflush(FILE *);
int printf(const char * __restrict, ...);
int scanf(const char * __restrict, ...);
```

Functions You SHOULD NOT Use

Variables

Employee payrollList[200] = {};

int payrollCount = 0;

double maxPay = -INFINITY, minPay = INFINITY, averagePay, totalPay = 0;

Includes

```
#include <stdio.h>

#include <string.h>

#include <math.h>
```

Possible Code to Use

```
fgets(payrollList[payrollCount].name, 200, payroll);
fscanf(payroll, " %c", &payrollList[payrollCount].title);
```

Passing in a string to a function

```
void optionA(char * string); // most preferred
void optionB(char string[]); // ok
void optionC(char string[100]); // ok
```

Passing in an array of strings to a function

void optionA(char ** string); // most preferred

void optionB(char * string[]); // ok

void optionC(char string[][100]); // warnings are annoying and common

- **COMMON WARNING: -Wincompatible-pointer-types**

void optionD(char string[100][100]); // warnings are annoying and common

- **COMMON WARNING: -Wincompatible-pointer-types**

Moral of the Story:

int ** is not equivalent to int *

They differ by a level of indirection. That is, int * [] is not int *.

This is another reason I prefer to do int * or int ** over int * [] or int [][], you can easily tell.

- Depending on your declaration and parameters, you may need to cast to the needed type

What is Wrong With This Code?

Define

```
struct myStruct {
    Int myInt,
    char * string,
    myStruct next,
    char type,
    int age,
} MyStruct;
```

Use

```
char * localStackStr = “Som”;
MyStruct local = .init();
local->myInt = 100;
->string = localStackStr;
->next = local;
MyStruct.age = 59;
MyStruct.type = “C”;
```

Due Dates

PA 6 – Tomorrow

Quiz 8 – November 13th (Monday)

Quiz 9 – November 17th (1 week from Monday)

PA 7 – November 29th (2 weeks, 6 days) Start once PA 6 is submitted!

PA 8 – December 8th (December 8th, you will have 1 week 2 days!) NO EXTENSIONS!

- I will provide less detailed feedback. I will only mark things that need to be addressed and provide brief reasons for lost points. Upon request, I will review it AFTER finals week and send you an email of your project.

Finals

Lab Final

- December 8th

Written Final (mostly multiple choice)

- Section **1: Tuesday December 12th**, 10:10 – 12:10 *in normal class room*

- Section **2: Thursday December 14th**, 08:00 – 10:00 *in normal class room*
  - If you do not like waking up by this time, the exam will not be fun, but you cannot pass w/o it

Note: Abnormal Slides were Intended to Standout.