PA 3 PUSHED TO MONDAY

EXAM COMMENTS

- Read the instructions
- Be sure you are getting help before the exam if you have questions
- Practice code on your own
  - If you are comfortable with another language, write code in that lang, then write it again in C/C++
- Be sure you give specific examples when it asks for when a data structure would be used
  - Unknown data size is correct, but not specific
- Arrays can grow or shrink when on the heap; growing and shrinking is very resource intensive

EXAM COMMENTS

- If you struggle with functions that I strongly hint at or explicitly state will be on the exam, see myself, Andy, or another TA before the exam
- Go to Martin's exam review; this will improve your understanding and readiness for exams
- Be sure to attend lecture the Wednesday before exams – you will likely be given an answer to one of the questions directly or indirectly

EXAM GRADING COMMENTS

- I decided to give points back for a certain criteria as I missed a few
    - Your grades in Canvas have been updated, so it may differ from what you see on your exam

- If you have questions, we will discuss
- If I made a mistake, your grade will be updated

- Average for lab is below 70%

LAB 4

General Feedback on PAs, Malloc

INITIALIZING A STRUCT INLINE

Suppose we have:

```
typedef struct _test {
    int first;
    char * second;
    double third;
} Test;
```

We can initialize as such:

```
Test myStruct = {
    .first = 987,
    .second = "Something",
    .third = 123.456
};
```

COMPARISON WITH NULL

- NULL is used with a pointer, not a value typically
  - In C, NULL used as a value makes it appear as if var is a pointer
- NULL is not intended to be used as a value for numerical values
- nullptr will represent a null pointer in C++
- NULL provides ambiguity since it can equal 0 or a pointer, so use nullptr in C++
  - Note: NULL will work the same in most cases, but nullptr is more specific and preferred

ENUM (EX 1)

- We can use an enum to declare a value without having to remember which number means what and reduces the amount of required comments

```
typedef enum : int {
    kEXIT, // Exit
    kEDIT, // Edit List
    kDELETE, // Delete From List
    kLOAD, // Load from the file
    kSAVE, // Save to the file
} UserSelection;
```

ENUM (EX 2)

- We can use an enum to declare a value without having to remember which number means what and reduces the amount of required comments

```
typedef enum : int {
    kOKAY, // Clean exit
    kFAILED_TO_OPEN_FILE, // Failed to open a file
    kFAILED_TO_CLOSE_FILE, // Failed to close a file
    kFAILED_TO_ALLOCATE, // Failed to allocate memory
} ErrorCode;
```

RETURN ERROR CODES

- Suppose a function could fail because memory is not allocated or the infile was not open
- When we return 1 or 0 (true false), we do not know why it failed
  - Therefore, we can return an error code
  - Check the value returned at the caller

FGETS

- Only use fgets, do not use fscanf, sscanf, or any of those horrible funcs
- Fgets will ensure we read the read the whole file
- Then, we use strtok to parse through the string
  - FIRST CALL: strtok(line, “,\n”)
  - OTHER CALLS: strtok(NULL, “,”)

MALLOC – WHEN WILL IT FAIL?

- On your system: probably never
- In the real-world:
    - When the device has limited memory
    - An attacker is maliciously sending a DOS attack
        - Fork-bomb (depletes memory and reduces availability of heap)
        - Many requests that use malloc
    - Using kernel space (kmalloc)

MALLOC FAILURE: EXAMPLE

- A new service has a single service since there are 100 users. The server has 100GB RAM. The system uses 30 GB on average.
- The service becomes popular and 10,000 users are now using the single server
- Suppose an average user consumes 7 MB after 1 week of usage
- @ 100 users, about 700MB is consumed
- @ 10000 users, about 7GB is consumed
  - This assumes all users consume 7 MB; if 0.5% (50 users) of users consume 1 GB, then we have:

$$50GB + (10000\ users - 50\ users) * 7\ MB = 69.96\ GB\ Consumed\ by\ Users$$
$$69.96\ GB + 30\ GB = 99.96GB; 0.04\%\ of\ \ memory\ remains$$

MALLOC FAILURE: EXAMPLE

- An attacker sends a fork bomb instruction to the server to run as an admin
  - Now, hundreds of thousands of processes are running in a matter of seconds
  - Windows: %0|%0
  - Linux: :(){ :|:& };:
  - C: int main() { while(1) fork(); }
  - Pearl: perl -e "fork while fork" &
- This will take a lot of memory and therefore reduces the memory available for malloc

LAB 4

LAB 4 - STACK

- A stack (programmatically) is like a singly linked list
    - Pointer to another node
    - Data
- Implement using a **singly** linked list

LAB 4 - TESTS

- We need tests for push, pop, isEmpty, top, peek
    - pushEmptyTest
    - pushNonEmptyTest
    - popEmptyTest
    - popNonEmptyTest
    - isEmptyEmptyTest
    - isEmptyNoEmptyTest
    - topEmptyTest
    - topNotEmptyTest

- Live Demo for writing peek test (if time)