A *Juicy* Overture of the C Programming Language

Instructor: Illustrious James “Tenebris” Theodore “Lux” Von Oswald The First, PhD (in progress), IEEE Ex-Chair (The Emperor), MD (probably not), CSS (stylesheet language)

Class Time: 8pm every day in Discord, office hours by appointment or DM
Class Structure: Webcams mandatory (because otherwise you will be asleep), begin by going over the homework, then move onto lesson.
Class Schedule:
Week 1: Basics
December 27th: Math Math Math Math
December 28th: Control Control Control!
December 29th: Pointers without tears.
December 30th: No Class, (Late Night Office Hours Available)
December 31th: No Class, New Years
January 1st: No Class, But Yes Office Hours
January 2nd: No Class, But Yes Office Hours
January 3rd: The Big Picture
January 4th: Strings and Things
January 5th: Enums and Unions
January 6th: James’s Bizarre Adventure (Miscellaneous features)
January 7th: Final Exam

Grading: If you can do the homework, you’re good, if not, you’re bad.
Final Exam: Linkedin Skill Quiz on C

Punishment for failing the class: Depression
Reward for Passing Class: IOU for an AI maid

Office Hours: DM me, I'm vibing 90% of the time and am happy to help out.

December 27th: Math, Logic, n' Bits

- Declaring variables
- Data Types
    - Numerics (char, int, float, double), casting (down and up)
    - Cv-qualifiers, Signed and Unsigned
    - Your first library, BOOL
- Operators
    - Math, Logic, Bit operations
- Extremely basic IO.

```c
#include<stdbool.h>
#include<stdio.h>

int main() {
    int x = 3;
    bool b = 0 && (x = 5);
    printf("%d", x);
}
```

Homework:

Question 1) Logic Operators

1.1) What does the following code on the right print out, and why isn't it 5?

1.2) Print out a truth table for demorgan's laws, show that ((not A) and (not B)) and (not (A or B)) evaluate to the same thing for all 4 possible combinations of A and B where A and B can be true or false.

Question 2) Bitwise Operations:

2.1) Print the 3rd bit (index 1 is the ones place) of the number 55 using a right shift and bitwise and with a mask.

2.1) Negate a number with two's complement using bitwise negation and + 1;

Question 3) Math Operators and Variables

3.1) What is the difference between i++ and ++i? Write a 2 line program that works differently when substituting an i++ with an ++i.

3.2) This is a famous problem given to highschool students when first learning a new language, it tests your mastery over variable declaration and math operators. Write a program to compute the date of Easter Sunday. Easter Sunday is the first Sunday after the first full moon of spring. Use the algorithm invented by the mathematician Carl Friedrich Gauss in 1800:

1. Let y be the year (such as 1800 or 2001)
2. Divide y by 19 and call the remainder a. Ignore the quotient.
3. Divide y by 100 to get a quotient b and a remainder c.
4. Divide b by 4 to get a quotient d and a remainder e.
5. Divide 8 * b + 13 by 25 to get a quotient g. Ignore the remainder.
6. Divide 19 * a + b - d - g + 15 by 30 to get a remainder h. Ignore the quotient.
7. Divide c by 4 to get a quotient j and a remainder k.
8. Divide a + 11 * h by 319 to get a quotient m. Ignore the remainder.
9. Divide 2 * e + 2 * j - k - h + m + 32 by 7 to get a remainder r. Ignore the quotient.
10. Divide h - m + r + 90 by 25 to get a quotient n. Ignore the remainder.
11. Divide h - m + r + n + 19 by 32 to get a remainder of p. Ignore the quotient.

Then Easter falls on a day p of month n.

For example, if y is 2001:

a = 6 b = 20 c = 1 d = 5 e = 0 g = 6 h = 18 j = 0 k = 1 m = 0 r = 6 n = 4 p = 15

Therefore, in 2001, Easter Sunday fell on April 15. Make sure you prompt the user for a year and have the user input the year. Also, make sure you output the values of p and n with the appropriate messages describing the values output.

December 28th: Control Control Control!

- Conditional control flow
    - If, if else, else if chains
    - Switch statements
    - Ternary operator
- Fixed Arrays
    - Without pointers for now.
- Loops
    - For, while, do while
- Functions
    - Pass by value

Homework:

Question 1): Functions

1.1) Construct a function that takes an input float, x, and an integer exponent, n, and computes and returns the float x^n.

1.2) Construct a void function that takes a char, c, and an int, n, and prints an n by n square made out of c to the console.

Question 2): Search Algorithms (Arrays, Functions, For Loops)

2.1) Construct a function that takes an array of 5 unsorted integers and a number to find, the function returns the index of the number to find in the array or -1 if the number is not found. Implement using linear search.

2.2) Construct a function that takes an array of 10 sorted integers and a number to find, the function returns the index of the number to find in the array or -1 if the number is not found. Implement using binary search.

Question 3): While Loops In action

Construct an algorithm that prints out all prime numbers up to infinity.

Stop crying, pointers are so easy, it's literally just where the variable live s.

December 29th: Pointers Without Tears

- Pointers
    - Pointer operators: address & and dereference *
    - Pointers as handles to arrays
    - Pointers as handles to strings
    - Pointer Arithmetic
    - Void pointers
- Advanced functions
    - Pass by reference with pointers
    - Multiple return values with pass by reference
    - Function POINTERS
- Dynamic memory allocation
    - Malloc, calloc, realloc and free.
- Structs
    - Literally just chunky boys
    - Member access (.), pointer member access (->)

Homework: Project 1, Making an ArrayList of ints in C

You will be creating a very bad dynamic arraylist "object" in C.

Your Arraylist will be a struct with two members:

1) the size of the arraylist as a size_t
2) A pointer to the dynamically allocated memory for the int array.

You should write the following "methods" as functions each method will take a pointer to the arraylist "object" it is operating on (fun fact, this is actually how C++ implements classes behind the scenes):

void createArrayList(ArrayList* list, size_t initialSize): set the initial size and allocate a chunk of memory of initialSize to hold our ints

size_t size(ArrayList* list): Get the current length of the array list.

int get(ArrayList* list, int index): return the element at the given index, print an error message if the index is out of bounds

void set(ArrayList* list, int index, int element): Write the value of the given element to the given index in the list. print an error message if the index is out of bounds.

void add(ArrayList* list. int element): Grow the arraylist by one element, use realloc to reallocate to resize the current array, set the end to the given element and update the size member.

void deleteArrayList(ArrayList* list): Free the dynamically allocated memory storing the array. The list list can no longer be used after this.

You must write test cases to ensure all of these "methods" work.

Bonus Problem 1: Reuse your code and write an arraylist for strings (const char*'s).

Bonus Problem 2: Write a singly linked list and write equivalents to the above functions for it.

January 3rd: The Big Picture

- Preprocessing directives
    - Define/undefine, ifdef, ifndef, else, endif, include, pragma
    - Include Guards
    - Special Preprocessing Operators (token concat operator ##) (stringifying operator #)
- Program structuring
    - Header Files, Forward Declarations, Implementation Files
- Compilation stages
    - Compiling, Object files, symbol tables, linking
- Running Under the hood, The C program inside of Memory
    - Sections: text, bss, data, stack and heap.

Homework:

Question 1) Define Macros

1.1) Write a define macro, MAX(A, B), that gives the max of two arbitrary numeric tokens A and B that evaluates to the value of the larger token.
1.2) Write a define macro, FOR(VARIABLE NAME, SIZE, STATEMENT), that generates typical for loops over a single statement with only the variable name, size expression, and statement to be run.

Question 2) Giving Project 1 Good Structure

Split up your implementation of project 1 into two files, A demo.c file that has your main function that runs your tests and An arrayList.c file that has your arraylist implementation (the struct and methods). Now write a header with forward declarations (for the struct and methods) for project 1 called arrayList.h. This header should have an include guard.

Your arrayList.c file should include arrayList.h
Your demo.c file should only include arrayList.h NOT arrayList.c.

Compile arrayList.c into an object file, arrayList.o.
Attempt to Compile demo.c into an executable, this should give an error, record the error and write a guess as to why this is happening.
Compile demo.c into an object file, demo.o. Notice that the error does not occur this time.
Link demo.o and arrayList.o into an executable and run it, it should work this time, Write a guess as to why the previous error is not occuring.

January 4th: Strings and Things

- We go over the entire <String.h> library
    - All the string functions, mem functions, etc
- We go over the entire <Stdio.h> library
    - Real printf, sprintf, scanf, etc.
    - File IO, fopen, fread, fseek …

Homework:
Pending

January 5th: Enums and Unions

We go over enums
We go over unions

Homework:
Pending

January 6th: James's Bizarre Adventure

We go over <math.h>
We go over the entire keywords list and see if there is anything we don't know
We go over vargs and define macro vargs.

Homework:
Pending

January 7th: Review and Final Exam

Good luck lmao.