Lab 5: Structures and Enumerations

CSE/IT 113

> "It was a beautiful, almost poetic way to cap what had been a textbook career of brilliant, original mathematical insights punctuated with outbursts of random, deeply unhinged violence."
>
> — Olin Shivers, Automatic Weapons and Computer Science

> "Computer Science is no more about computers than astronomy is about telescopes."
>
> — Edger Dijkstra

---

1 Introduction

This lab focuses on arrays of structures. If you need help with this assignment, please visit the help hours! The schedule is posted on Canvas.

2 Zombie Apocalypse

This week you will write a program to record a *zombie apocalypse log*.

The log allows you, in the event of an actual zombie apocalypse, to keep track of the zombies that you encounter. You will enter whether you encountered a dead zombie or one that was still alive. If you encounter a dead zombie, you will be asked to take note of how many toes it had, and if if was alive, you can record how many milliliters of blood oozed from its body after you killed it.

You will also record the day and time that you encountered this zombie.

For this lab, you will be using a structure that looks like this:

```
struct zombie{
   char dead;  /*y if dead, n if alive*/
   enum {MONDAY = 1, TUESDAY, WEDNESDAY, THURSDAY,
         FRIDAY, SATURDAY, SUNDAY} day;
   int toes;
```

```
    float blood;
    int hour;
    int min;
    int sec;
};
```

Your program should display the following menu when run:

```
1) Enter new zombie information
2) Display zombie information
3) Return to fighting zombies (exit)
>>
```

If you type 1 and hit enter, your program prompts you like this:

```
Was the zombie found dead? Y or N
>>
```

You can only enter one or the other because the memory we are using is very expensive. We only have room for one. At this point you must enter either Y or N. Then a corresponding question will follow. This is an example of the code running.

```
1) Enter new zombie information
2) Display zombie information
3) Return to fighting zombies (exit)
>> 1

Was the zombie found dead? Y or N
>> Y

Please enter the number of toes the zombie has.
>> 34

Please choose the day this zombie was encountered:
1) Monday
2) Tuesday
3) Wednesday
4) Thursday
5) Friday
6) Saturday
7) Sunday
>> 5

Enter time when this zombie was encountered.
Separate hours, minutes, and seconds by colons.
HH:MM:SS
>> 12:33:42
```

```
1) Enter new zombie information
2) Display zombie information
3) Return to fighting zombies (exit)
>> 1

Was the zombie found dead? Y or N
>> N

Please enter the amount of blood that oozed from its body
after you killed it (in mL)
>> 73.32

Please choose the day this zombie was encountered:
1) Monday
2) Tuesday
3) Wednesday
4) Thursday
5) Friday
6) Saturday
7) Sunday
>> 1

Enter time when this zombie was encountered.
Separate hours, minutes, and seconds by colons.
HH:MM:SS
>> 01:22:12

1) Enter new zombie information
2) Display zombie information
3) Return to fighting zombies (exit)
>> 2

1. This zombie was found dead.
This zombie had 34 toes.
This zombie was sighted Friday at 12:33:42.

2. This zombie was found ALIVE!
It was drained of 73.32 mL of blood once killed.
This zombie was sighted Monday at 01:22:12.

1) Enter new zombie information
2) Display zombie information
3) Return to fighting zombies (exit)
>> 3
```

```
Stay alert! Keep a watch out for zombies!!

GOODBYE and GOOD LUCK!
```

Notice how when you print out the data, you print either blood or toes. You will need to account for this since the structure contains both variables. (Hint: Use char dead)

For this lab, you should only support having five entries (hint: use an array). However, the array is circular, meaning on the 6th entry, it is place in the first element of the array, etc. You do this to save memory. (HINT: To do this you will use the modulo operator).
For this lab, you are required to:

- Implement the user interface described above.
  - Read the time stamp from the console – three integers separated by colons – using `fgets` and `sscanf`.
- Use an array of structures to hold five data collections in a circular buffer.
- Use an enumeration to keep track of which day of the week the zombie was found on
- Use a structure to hold all of the zombie "stats" (see example above)
- Use `fgets` and `sscanf` for input. **No use of scanf is allowed!**
- Except for a few variable declarations, a few `fgets` statements and some `switch` statements for the menu logic, the majority of the code in `main` should be function calls.
- These functions are required for your lab.
  - A **get_time** function to get the day and the time
  - A **get_toes** function to get the number of toes the dead zombie has
  - A **get_blood** function to get how much blood oozes from un undead zombie once it has been killed.
  - A **print_data** function that displays all of the zombie information you have entered into the database
- Like last week, you need to implement a header file for this lab. For this week's lab, your structure will go in `zombie.h` along with your function prototypes. All of your function implementation will go in `zombie.c` Your main should be in `lab5.c`. You compile in a similar fashion as in lab 3.

3 Additional Problems

- Problem 1 from Project Euler:

  If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Find the sum of all the multiples of 3 or 5 below 1000.

Name your source code `multiples.c`. Capture the output in a script file named `multiples.script`.

- Problem 5 from Project Euler:

  2520 is the smallest number that can be divided by each of the numbers from 1 to 10 without any remainder.

  What is the smallest positive number that is evenly divisible by all of the numbers from 1 to 20?

  Name your source code `smallest.c`. Capture the output in a script file named `smallest.script`.

- Problem 6 from Project Euler:

  The sum of the squares of the first ten natural numbers is,

  $1^2 + 2^2 + ... + 10^2 = 385$

  The square of the sum of the first ten natural numbers is,

  $(1 + 2 + ... + 10)^2 = 55^2 = 3025$

  Hence the difference between the sum of the squares of the first ten natural numbers and the square of the sum is 3025 385 = 2640.

  Find the difference between the sum of the squares of the first one hundred natural numbers and the square of the sum.

  Name your source code `difference.c`. Capture the output in a script file named `difference.script`.

- From Project Euler, Counting Sundays:

  You are given the following information, but you may prefer to do some research for yourself.

```
1 Jan 1900 was a Monday.
Thirty days has September,
April, June and November.
All the rest have thirty-one,
Saving February alone,
Which has twenty-eight, rain or shine.
And on leap years, twenty-nine.
A leap year occurs on any year evenly divisible by 4,
but not on a century unless it is divisible by 400.
```

  How many Sundays fell on the first of the month during the twentieth century (1 Jan 1901 to 31 Dec 2000)?

  Name your source code `sundays.c`. Capture the output in a script file named `sundays.script`.

Some hints:

You can use the `cal` command to display the calendar any year, e.g. `$ cal 1900`.

You can determine if a year is a leap year by using the following:

```
if (year % 4 == 0 && year % 100 != 0) || year % 400 == 0)
       /* leap year */
else
       /* not a leap year */
```

As a way to get started: manually figure out, using `cal`, how many Sundays fell on the first of the month between 1901 and 1905 and write a program to solve that. After that program works correctly extend it to the year 2000.

4 Submission guidelines

Submit the tar archive containing your README, source code, and script files to Canvas before the due date. Your tarball should contain the following files:

```
lab5.c
zombie.c
zombie.h
multiples.c
smallest.c
difference.c
sundays.c

lab5.script
multiples.script
smallest.script
difference.script
sundays.script
```

Name your tarball `cse113_firstname_lastname_lab5.tar.gz`.