Lab 4

Instructions: Use functions and function calls to implement digit-based functions.

Objectives:

- Explore functions and function calls.
- Design procedures to manipulate the digits of an integer variable.

Task 1: Implement the following functions in a C program called `Lab4a.c`:

- `int numDigits(int x)`: Returns the number of digits in x. Examples:
  - `numDigits(1)` → 1
  - `numDigits(137)` → 3
  - `numDigits(13794)` → 5
- `int getDigit(int x, int i)`: Returns the *i*th digit of x. Examples:
  - `getDigit(13794, 1)` → 4
  - `getDigit(13794, 2)` → 9
  - `getDigit(13794, 3)` → 7
  - `getDigit(13794, 4)` → 3
  - `getDigit(13794, 5)` → 1
- `int sumDigits(int x)`: Returns the sum of all digits in x. Examples:
  - `sumDigits(1)` → 1
  - `sumDigits(11)` → 2
  - `sumDigits(44)` → 8
  - `sumDigits(12345)` → 15

In the `main` function, use `scanf` to prompt the user for `x` and `i`, then print the results of the three functions as shown in Figure 1.

```
/home/user/CIS190/Lab4/$ ./Lab4a.out
Enter a number: 13794
Which digit? 2
numDigits(x = 13794) = 5
getDigit(x = 13794, i = 2) = 9
sumDigits(x = 13794) = 24
/home/user/CIS190/Lab4/$ ./Lab4a.out
Enter a number: 54321
Which digit? 4
numDigits(x = 54321) = 5
getDigit(x = 54321, i = 4) = 4
sumDigits(x = 54321) = 15
```

Figure 1. Example outputs for `Lab4a.c`.

Hints:

- You may assume `x` is a nonzero integer, and that `i` is a nonzero integer within `x`'s size. (No error checking required for these conditions.)
- Consider how `numDigits` and `getDigit` might be useful for implementing `sumDigits`.

Task 2: In a separate C file Lab4b.c, modify `numDigits`, `getDigit`, and `sumDigits` follow these new specifications:

- `void numDigits(int x, int *ret)`
- `void getDigit(int x, int i, int *ret)`
- `void sumDigits(int x, int *ret)`

Rather than return results with the `return` keyword, these modified functions should pass a variable `ret` by reference and store results in this variable before terminating.

Hint: The output for this program should be the same as shown for Task 1 in Figure 1. Only the internal implementation should change.

Submission details:

- Upload a compressed archive (e.g., .zip) containing `Lab4a.c`, `Lab4b.c`.
- The archive should be named `Lab4_LastName`, where `LastName` is your last name.
- If you're on Linux, you can use the following command to create a .tar.gz archive from the terminal:

  ```
  $ tar -czvf Lab4_LastName.tar.gz Lab4a.c Lab4b.c
  ```

  where `LastName` is your last name.