CPSC 320: Assign 0
Linux and C Warmup

Objective:
This assignment is intended to get you flowing with the Linux environment and C programming. These are mostly CS I caliber programs, but with a C twist. You should already be familiar with the concepts and be able to focus on the transition.

Exercices:

1. **Basic input/output and conditionals**
   Create a program that prompts the user for their name and age. The program should print the difference between your age and the input. The words ”younger”, ”older” or ”same age” should be used in the output, depending on the difference in age.

   Example output:

```
$ ./ages
What is your name? Steve
How old are you, Steve?  29
You are 11 years older than me.

$ ./ages
What is your name? Jane
How old are you, Jane?  15
You are 3 years younger than me.
```

2. **Loops**
   - **a.** Write a program that asks for a number. Then the program should print 1 through the given number on separate lines.
   - **b.** Encapsulate your code in a while-loop that asks the user if he/she would like to run the program again. Note that when reading a character from the input stream, part of the newline from the previous input is still buffered and considered as input. To discard the newline, start the scanf string with a space like this: scanf(" %c", &input);.

   Example output:

```
$ ./loops
Give a number: 5
1
2
3
4
5
Run again (y/n)? y
Give a number: 2
1
2
Run again (y/n)? n
Done...
```

3. **Arrays**

Write a C function that calls sub-functions to:

- **a)** Count the number of 0's in an integer array. The number of 0's is returned by the function.
- **b)** Print an array of integers. The integers are printed on one line, enclosed in curly brackets and separated by a coma.
- **c)** Triple the value of all elements in an array of integers

Each sub-function take two parameters, a pointer to the array of integers and the number of elements in the array. Write a main function that declares and defines an array of integers, containing at least 10 integers. Use your other functions to print the initial array, the number of zero-valued elements in the array and the contents of the array when all elements have been tripled.

Example output:

```
$ ./arrays
Initial array: { 1, 2, 3, 0, 5, -6, 0, -8, 0, 10 }
Number of 0's: 3
Tripled array: { 3, 6, 9, 0, 15, -18, 0, -24, 0, 30 }
```

4. **Command Line Arguments**

Write a program that outputs a multiplication table. The program takes 2 optional (for the user, not for you) arguments: input file and output file. 9 They are specified with -in and -out . The order should not matter. If no input file is specified, stdio is used as input. If no output file is stdout is used for output. The user specifies number of rows and columns, in that order, for the multiplication table with two integers. The columns must be aligned for all values not exceeding 1000. Remember to close any files that you opened.

Example output:

```
$ ./multiply
 4 5
1       2       3       4        5
2       4       6       8       10
3       6       9      12       15
4       8      12      16       20

$ ./multiply -in input.txt
1       2       3       4       5       6
2       4       6       8      10      12
3       6       9      12      15      18
4       8      12      16      20      24
5      10      15      20      25      30
6      12      18      24      30      36
7      14      21      28      35      42
8      16      24      32      40      48
9      18      27      36      45      54
10     20      30      40      50      60

$ ./multiply  -out answers.txt -in input.txt
```