ECEN 324
Lab #1 – C Programming

This lab assignment is to practice simple C-language coding as a review. It is to be completed individually as "Open Book no Solutions." **Absolutely no A.I. should be used at any point in completing this assignment.** The C program is to produce exactly the same output as the provided solution file, including whitespace.

Getting the starting file:

The C program that you will be completing may be found on the ECEN 324 Linux VM:

```
/home/ecen324/cLab/ecen324_c_lab.c
```

You might copy it with a command like the following to your own home directory (or possibly a subdirectory you have made in your home directory):

```
cp /home/ecen324/cLab/ecen324_c_lab.c lab1.c
```

Compiling your finished program:

Your C code should be compiled with `gcc`. That command might look like the following, if you name the output executable as `mylab1`:

```
gcc -o mylab1 lab1.c
```

Comparing the output of your program with the solution file:

To check the correctness of your program, use "`55`" as the input value the program should ask for and use the "`diff`" command to compare your output with the `c_lab_solution.txt` file in `/home/ecen324/cLab/`.

Here is an example of a successful correctness check where the diff command sees no differences between the solution file and the output of the compiled program:

```
[me@jordanvm c_lab]$ gcc -o mylab1 lab1.c
[me@jordanvm c_lab]$ mylab1 >mylab1.out
Enter number: 55
[me@jordanvm c_lab]$ diff /home/ecen324/cLab/c_lab_solution.txt mylab1.out
[me@jordanvm c_lab]$
```

The lack of output after the `diff` command indicates that the two files were identical.

The ">" in the second command line above, redirects standard output generated by `printf()` calls of the `mylab1` executable to the file named `mylab1.out`. The reason the "`Enter number:` " prompt does not get redirected to the `mylab1.out` file, is that the prompt should be printed with an `fprintf()` command and sent to standard error.

If the `diff` command gives any output and says the outputs differ, but to you the files look identical, there is probably a problem with whitespace. If it is only a problem with whitespace, such as an extra space, or using spaces instead of a tab in the table, running a `diff` command with the `-w` option will show the files as being the same. Using the `-w` option with the `diff` command identifies for sure that it is a whitespace issue if the files are different. To learn about the diff command and its options (like: --color,) you might use the `man` or `info` commands. You might also see: https://linuxhandbook.com/diff-command/.

As another tip for catching whitespace differences, if you "pipe" (|) the `diff` output to the `cat` command with the `-et` flags, it will show end of lines (`-e`) and tabs (`-t`) with special characters. Example:

```
diff file1 file2 | cat -et
```

Submitting your .c file:

Submit your C code by executing the "submit" command, like:

```
submit lab1.c
```

The header in the `.c` has the proper lab name and format for use with the submit command. You should edit the header with your name and appropriate comments.

C language helps:

These C programming tutorials might be helpful:

- https://computer.howstuffworks.com/c.htm/printable
- https://www.tutorialspoint.com/cprogramming/index.htm

This C reference card might also be useful: https://users.ece.utexas.edu/~adnan/c-refcard.pdf

The `man` and `info` commands may be used to access documentation on UNIX/Linux commands, libraries, file formats and other items. There are various *sections* to the "manual" pages and doing a command like:

```
man printf
```

The above command might bring up information in section 1 (one) of the man pages where what you really wanted was the information in another section. To get to the documentation on printf that will be most helpful, try:

```
man 3 printf
```

Other notes:

You will lose points if you print each line of the body of the table with its own `printf()` statement. Do something more elegant than that, specifically a loop.

For dynamic memory allocation in C, you have to check that the `malloc()` didn't return a null pointer. If it does, it means the malloc was unsuccessful because the system is out of memory.

Properly comment your code. You should not leave the comments in your submitted code that say "add code here". Add comments that describe how you created your table. Add comments that describe your malloc checks.