Homework #1
EC535, Fall 2025
Assigned: Sept 8, Monday -- **Due:** Sept 16, Tuesday, 9 PM

This homework is a warm-up exercise of **C programming** under Linux environment. Your submission MUST work on lab machines. Please make sure to test as needed. **Code that is not compiling on lab machines will not receive credit.**

Remote access instructions are posted on Piazza. Note that if you are connecting from off-campus, you will need to use **VPN first.*

---

0. Many of the future assignments will require basic knowledge of make. A good reference for writing makefiles can be found at: http://www.gnu.org/software/make/manual/make.html. Study the reference, and create a bookmark in your browser to the link for future reference.

1. Write a **C program** that reads from an input file a list of *unsigned* 32-bit decimal integers, and prints *two columns* of integers to an output file.

   The first output column contains the binary mirrors of the input numbers **in decimal**. For example, the 32-bit binary representation of $19088743_{(10)}$ is $0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111_{(2)}$. Its binary mirror is $1110\ 0110\ 1010\ 0010\ 1100\ 0100\ 1000\ 0000_{(2)}$, which is $3869426816_{(10)}$.

   The second output column contains the number of "010" sequences in the binary representation of each original (not-mirrored) input number. For the same example above, the binary representation of $19088743_{(10)}$ contains 5 instances of "010" (00000001001000110100010101100111$_{(2}$. So, the second output column for this number should be 5.

   Your program should consist of three files: main.c, bits.c and bits.h. You main.c file contains the main function and processes file I/O. Your bits.c and bits.h files should define two functions:
   ```
   unsigned int BinaryMirror(unsigned int);
   unsigned int CountSequence(unsigned int);
   ```

   The first function computes the binary mirror and returns the number in an unsigned int. The second function counts the number of "010" sequences. The resulting executable should be named "**MyBitApp**". The program takes the name of the input file as its first command line argument and the output filename as the second argument, such as:
   ```
   >    MyBitApp myinput.txt  myoutput.txt
   ```
   (Your executable should be able to work with arbitrary input and output file names.)

2. Implement a linked list to maintain a prioritized list of items, where each item may include the input number, binary representation, and the ASCII representation (you will determine the contents yourself). In the output file, the information printed should be *sorted based on the ASCII representation (in ascending order)*.

   Note: This feature can be implemented via arrays as well; however, implementing a linked list is required for receiving credit on this question. Check out online links or printed resources on *when to use linked lists vs. arrays*. Please place your linked list in mylist.c and mylist.h.

3. Write a Makefile to compile your program from parts 1 and 2. The Makefile should compile **only the necessary files** after an update of the source code. For example, if only main.c is updated, the Makefile should only compile main.c but not bits.c or mylist.c; but if bits.h is updated, all .c files that include it should be recompiled. The default (first) rule should produce the "**MyBitApp**" executable. Provide a *clean* rule in the Makefile which will remove all object files (and executables) generated by running make.

   Use the **"gcc"** command to compile your program. If you are not familiar with gcc, type "man gcc" to learn its command line options. Please write your code cleanly and include comments.

   Please include your name and BU username as a comment at the beginning of each file.

Reminders

- You must work on your own. You can discuss general ideas with others, search online resources or books, or use AI tools while studying (e.g., "can you show me an example of …" type prompts are particularly helpful), but **you must NOT look at other students' code, NOT share your code, and NOT submit AI-generated code.** This course uses automated software for plagiarism checking.
- *Recall the late penalty policy:* Late submissions receive 20% reduction of the grade for up to 48 hours after the deadline. Submissions that are delayed for more than 48 hours are not accepted.

Submission Instructions

- Make sure to follow the guidelines for function names, executable names, folder naming conventions, etc. not to lose any points. We make use of scripts and software while grading for efficient and consistent grading. So it is essential for everyone to follow the given guidelines.
- Put your **source files (main.c, bits.c, bits.h, mylist.c, mylist.h) and the Makefile** in a folder called YourBUusername_HW1. Do not include any other files. Submit a single compressed archive, YourBUusername_HW1.zip, on GradeScope.
- You can submit multiple times. However, keep in mind that we will always grade your latest submission and will discard any earlier submission from the same student. *E.g., if you have a late submission, that submission will be graded and all other earlier submissions will be discarded.*
- If you are coding in an environment other than the PHO307/305 lab machines, MAKE SURE your code compiles and works before submitting. Assignments that do not compile WILL NOT receive credit, no exceptions.
- Tentative rubric:
  - Makefile: 20 points
  - Main.c and I/O handling: 20 points
  - Bit manipulation: 40 points
  - Correct implementation and use of linked lists: 20 points