Homework #1
EC535, Spring 2019
Assigned: Jan 24, Thursday -- **Due:** Jan 31, Thursday, 10:00pm

This homework is a warm-up exercise of **C programming** under Linux environment. Use the **PHO 307 (or 305)** lab computers for this homework OR access the engineering grid:
eng-grid.bu.edu / eng-grid2.bu.edu via **ssh**.

Note that if you are off-campus, you will need to use **VPN to access the grid.*

0. **In class component (January 24):**
   Update the C code you are given (hw1intro.c) such that the "myNewThread" function asks the user to enter an integer and then prints out this integer in decimal, binary, and hexadecimal form.

   Submit by 10:45am two files ONLY: hw1intro.c and hw1intro (executable) via email to ec535spring2019@gmail.com, with subject line "YourBUusername_HW1inclass".

---

1. Many of the future assignments will require basic knowledge of make. A good reference for writing the Makefile 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.

2. Write a C program which 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 000**1** **0**0**10** 001**1** **0**1**0**0 0**101** **0**1**10** 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 "10" (i.e., a "1" followed by a "0") sequences in the binary representation of each original (not-mirrored) input number. For the same example above, the binary representation of $19088743_{(10)}$ contains 7 instances of "10". So the second output column for this number should be 7.

   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 SequenceCount(unsigned int);
   ```

   The first function computes the binary mirror. The second function counts the number of "10" sequences. The resulting executable should be named "**BitApp**". It takes the name of the input file as its first command line argument and the output filename as the second argument, such as:
   ```
   >    BitApp myinput.txt  myoutput.txt
   ```
   (Your executable should be able to work with arbitrary input and output file names)

3. Implement a linked list to maintain a prioritized list of the items (i.e., 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)*. This feature can be implemented via arrays as well; however,

implementing a linked list is required for getting credit on this question. Check out online links or printed resources on *when to use linked lists vs. arrays*.

4. Write a Makefile to compile your program from parts 2 and 3. 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; but if bits.h is updated, all .c files that include it should be recompiled. The default (first) rule should produce the "**BitApp**" 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 as a comment at the beginning of each file.

Reminders

- *Reminder of the collaboration policy:* You should work completely on your own. You can discuss general ideas with others but **you should NEVER look at other people's code and never share your code. This course uses automated software for assignment similarity checking.**
- *Recall the late penalty policy:* Late submissions receive 5% reduction of the grade every 6 hours. Submissions that are delayed for more than 48 hours after the deadline are not accepted.
- You should only submit the in-class component by 10:45am on January 24, Thursday. No late submissions.

Submission Instructions

- Make sure to follow the guidelines for function names, executable names, folder naming conventions, etc. not to lose any points. As the class includes a large number of students, we make use of scripts and software while grading. So it is essential for everyone to follow the given guidelines.
- Put your **three source files and the Makefile** in a folder called YourBUusername_HW1. Do not include any other files. Submit a single compressed archive, YourBUusername_HW1.zip, via email to ec535spring2019@gmail.com, with a subject line YourBUusername_HW1.
- 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 eng-grid or lab machines, MAKE SURE your code compiles and works before submitting. Assignments that do not compile WILL NOT receive credit.
- 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
    - In-class exercise: 5 points extra credit