Computer Security (CS:4640)

Homework 1 (Warmup) Total points: 100

Due: 1 Week (by 11:59 pm of January 28, 2017)

This assignment will contribute 5% to your final grades.

Homework Type: Individual programming assignment.

Objective: The objective of this homework is to introduce you to binary, decimal, and hexadecimal numbers. Another major objective of this homework is to introduce students to reading and manipulating binary data stored in a file. Being able to comfortably manipulate binary data is paramount in cryptography.

Programming Languages: For this homework, the allowed programming languages are C/C++, Java, and Python.

Library Usage: Please do not use any library, API, or language features to directly perform the base conversions (e.g., `printf` with %x format specifier).

Submission: Please submit your source code through ICON by respecting the following instruction. The top level directory of your submission should be named
⟨last-name⟩-⟨first-name⟩-Homework-1. If a student's name is Bob Marley and he were to submit the assignment, his top level folder name will have the name `Marley-Bob-Homework-1`. Inside the top level directory you will have two folders with names "Problem-1" and "Problem-2", respectively. All the source code for problem 1 should reside in the directory "Problem-1" and all the source code for problem 2 will be inside the folder "Problem-2". In addition to the source code inside the folder "Problem-1" (resp., "Problem-2"), you should also include a file named README which specifies how to compile your source for problem 1 (resp., problem 2). You will then zip your top-level directory (e.g., `Marley-Bob-Homework-1.zip`) and submit it through ICON.

Grading: Please make sure your source code compiles, otherwise you will not obtain any points. Please comment your source code generously so that it is easy for the TA to understand it. Your program will be graded automatically against some TA-generated test cases. **Please follow the instructions accurately to avoid getting a zero**.

Cheating and Collaboration: *This is an individual project, you can discuss with your peers but cannot copy source code. Please do not copy source code from Internet.*

1. (60 points) (**PROBLEM 1: Base Conversion**) You are to write a program for performing base conversions between binary numbers (base-2), hexadecimal numbers (base-16), and decimal numbers (base-10). Your program should take the following command line arguments: (1) A string whose value can be one of the following: `bin2hex`, `hex2bin`, `dec2hex`, and `dec2bin`; (2) A string denoting the input file name f_i; (3) A string denoting the output file name f_o.

   Each hexadecimal number can only have the following characters: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. Each hexadecimal number should be preceeded by `0X`. For instance, `0XABC1` is a valid hexadecimal number whereas `0XVV0` or `ABC1` are not.

   ## Options:

   Option `bin2hex`—This option tells your program that the input file f_i contains several binary numbers in ASCII representation and your program should convert these numbers to their equivalent hexadecimal numbers and print it in the output file f_o in their ASCII representation. If the input binary number is `0000_1111` then the corresponding hexadecimal number is `0X0F`. If the binary number is `0000_0000` then the corresponding hexadecimal number will be `0X00`.

   Option `hex2bin`—This option tells your program that the input file f_i contains several hexadecimal numbers in ASCII representation and your program should convert these numbers to their equivalent binary numbers and print it in the output file f_o in their ASCII representation.

   Option `dec2bin`—This option tells your program that the input file f_i contains several decimal numbers in ASCII representation and your program should convert these numbers to their equivalent binary numbers and print it in the output file f_o in their ASCII representation. If the decimal number is 10, then the corresponding binary number is `00000000000000000000000000001010`.

Option `dec2hex`—This option tells your program that the input file f_i contains several decimal numbers in ASCII representation and your program should convert these numbers to their equivalent hexadecimal numbers and print it in the output file f_o in their ASCII representation. If the decimal number is 10, then the corresponding hexadecimal number is `0X0000000A`.

This problem does not require you to convert hexadecimal or binary numbers to decimal numbers.

Description of the Input/Output file Format

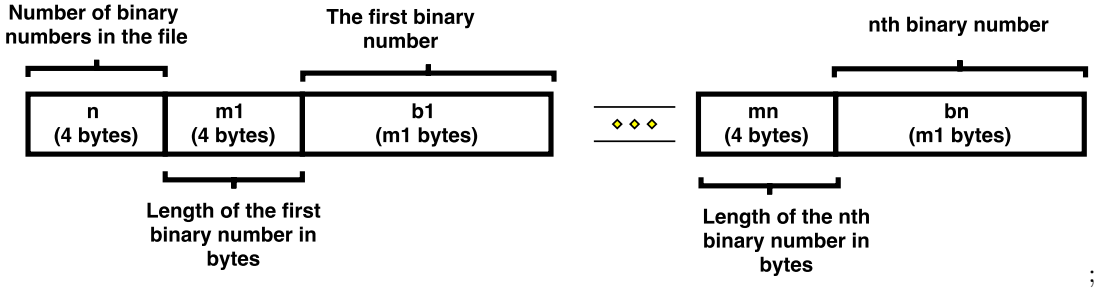
Input and Output file format for binary numbers in ASCII representation: The first line of the (input/ouput) file is a non-negative integer n. Then n lines follows where each of which contains a description of a binary number. Each of the n lines has the form $l\langle\text{space}\rangle b$ ($\langle\text{space}\rangle$ is the actual space character) where b is the binary number whereas l is a non-negative integer representing the number of digits (i.e., 0 or 1) in the binary number b. Note that, l will always be a multiple of 4 (i.e., $l\%4 = 0$).

Input and Output file format for hexadecimal numbers in ASCII representation: The first line of the (input/output) file is a non-negative integer n. Then n lines follows where each of which contains a description of a hexadecimal number. Each of the n lines has the form $l\langle\text{space}\rangle h$ where h is the hexadecimal number whereas l is a non-negative integer representing the number of digits in the hexadecimal number b including the preeceding `0X`.

Input file format for decimal numbers in ASCII representation: The first line of the file is a non-negative integer n. Then n lines follows where each of which contains a decimal number. Each of the n lines has the form d where d is the decimal number that needs to be converted.

You will be given three sample files, namely, `decfile`, `binfile`, and `hexfile`, and they represent how decimal, binary, and hexadecimal numbers are stored in a file.

2. (40 points) (**PROBLEM 2: Dealing with Binary Data**) In this problem, you are to write a program that takes an input binary file name f_{in} and an output file name f_{out} as command line arguments. The input file f_{in} will contain several binary numbers and your program should write their hexadecimal equivalent to the output file f_{out}. **The specific format of input and output files are given below**.



;

Input file format (Binary): The first 4-byte of the input binary file represents a non-negative number n where n denotes the number of binary numbers in the input file. For each of the n binary numbers, the first 4-byte denotes a non-negative number m where m represents that the binary number consists of m bytes. Following the 4-byte representing m, m bytes will follow—each of which you should consider as a non-negative number—denoting the binary number b. Your goal is to convert b to its hexadecimal representation and print it in ASCII in the output file. See the figure above for details.

Sample input file: You are given a sample binary input file, named `bin_in`, as part of this assignment. Note that if you open the file with a text editor it might display jibberish because the file is in binary, not ASCII.

Output file format (ASCII): The first line of the output file is the number n that you obtained from the input file. Next, n lines will follow; one for each binary number in the input file. Each of these n ouput line will be of the form $l\langle\text{space}\rangle h\backslash$n (here, $\langle\text{space}\rangle$ denotes an actual space character whereas \n denotes the new line character). In $l\langle\text{space}\rangle h\backslash$n, h is the hexadecimal representation of the input binary number preceeded with a `0X` and l denotes the number of characters used in h including the preeceding 0X.

If you have a one-byte binary number `0000_1111`, the corresponding output line should be the following:

$4\langle\text{space}\rangle 0X0F\backslash$n

Sample output file: You are given a sample output file `hex_out` corresponding to the input file `bin_in` as part of this assignment. Note that, you can view the output file in a normal text editor.

Testing your program: To test your program, you are given a sample binary input file generator program named `inputGen.c` which takes a file name f as a command line argument and generates a random binary file with the name f. To compile the input file generator, type in `gcc inputGen.c -o ingen` in your terminal. After compilation, for executing the input generator to generate an input file named `test.bin`, you need to type `./ingen test.bin` in your terminal.