Introduction to C Programming I

Summary: In this homework, you will implement a set of programs to practice working in the C language under Linux.

1 Background

In this assignment you will start to learn a new programming language (C), and try doing program development on Linux. You will work on a set of programs that introduce the basic syntax of C, the structure of programs, scoping rules, and data types. Since C is a new programming language for many students, the programs you will be writing in this assignment will be a bit simpler than the programs you are used to developing. We ask that you do your program development on GCC compiler under Linux. For this first assignment, you will use the NetBeans IDE to access the GCC compiler.

In previous courses, you mainly used the Java programming language. In other courses you may have been briefly introduced to other languages like Prolog or C++. For this course, we will be using C. The most significant difference between C and other languages you have in the past, is that C is an imperative language, not an object-orientated one. In a language like Java, you created classes which contained both state and behavior. In C, there is no concept of classes (or objects). Instead, programs consistent of a set of functions. A function is analogous to a method but does not exist in the context of a class. It is simply a named piece of code with specific input and output. C programs are developed by writing functions and then composing them together. C is also different than Java because it is a low level language. Unlike Java, C programs have the ability to directly interface with a computer's hardware (e.g., memory via pointers). This is the reason we will be learning C - it is a common language to implement OSs because OSs need the ability to manage the low-level resources of a computer and allocate them to different programs.

For this unit, there is a set of course videos which describe how to install Xubuntu and NetBeans on a virtual machine. By following those instructions, you will be able to run a "virtual" instance of Xubuntu (a variant of the common Ubuntu Linux distribution) on your computer. NetBeans is multi-language IDE you may have used in the past. For this assignment, you should plan to use it to develop your C programs. Note that IDEs aren't as common on Linux as other operating systems. Many developers write code primarily from a "command prompt" environment called the terminal. This is to be discussed at a later time.

In this assignment, you will work on three programs. In the first, you will debug an existing program. Some of the bugs that this program contains are not issues that can arise in a language like Java. In the second, you will develop a program to read data from the user and preform a computation based on what they enter. In the third, you will develop a program that manipulates C-style strings to decrypt a message.

This document is separated into three sections: Background, Requirements, and Submission. You have almost finished reading the Background section already. In Requirements, we will discuss what is expected of you in this homework. Lastly, Submission discusses how your source code should be submitted on Black-Board.

2 Requirements [30 points]

For this assignment you will write a series of three small programs to practice programming in C. **Your programs must compile and run under Xubuntu (or another variant of Ubuntu) 18.04.** If you follow the video sequence on installing NetBeans for C/C++ for Xubuntu, you will have this development environment available. For two of these programs, you must start from the attached base files. For the other program, you will be expected to write it from scratch.

2.1 Computation (hw02a_base.c) [8 points]

For this program, you will write a program to compute the total volume for a number of cylinders. A cylinder may have different values for height and radius. Study the base code and make the following changes:

- Add a prompt for the number of cylinders to sum. Make sure to use the control symbol for integers. [2 points]
- Create a loop based on the number of cylinders the user enters. [1 points]
- Within the loop, prompt for height and radius parameters (allow floating point numbers), and calculate the volume for that particular cylinder. [4 points]
- Display the total volume sum back to the user. Make sure to use the right control symbol. [1 points]

2.2 Collatz (no base file) [8 points]

For this program, consider the following pseudocode algorithm for checking if a particular positive integer satisfies the collatz conjecture:

Algorithm 1 Collatz conjecture checker.

```
termination(n) { //pre-condition: n is any integer
  while n not equal 1 do {
    if even(n)
    then n := n/2
    else n := 3n+1
  }
  output(n) // post-condition: n=1
}
```

- Write a C program to implement the algorithm as a function called *termination* that is called from main. [4 points]
- Call *termination* from main on an input integer given by the user. [2 points]
- Count and display how many iterations are executed for the input. [2 points]

2.3 Decryption (hw02b_base.c) [14 points]

For this program, you will write a chunk based decryption program. Study the base code and make the following changes:

- Implement display_chunks(). Loop over each string in the array of message chunks (strings) and print it. Each chunk should be printed back to back, do not insert an extra newline. Do not print the first character in each string since it is only used to store the order of the chunks. (Hint: use pointer arithmetic to skip a character.) [2 points]
- Implement decrypt_chunks(). Loop over each string in the array and shift the characters in it by subtracting DECRYPTION_SHIFT value from them. Use pointer arithmetic to access individual characters but array access to the strings. Remember that C-style strings have a null terminator at the end. Do not apply the shift to the terminator. (Hint: to avoid doing double pointer arithmatic, save a char* pointer to the active chunk[?] in the outer loop but before the inner loop. Then the inner loop is only concerned with a single array of characters rather than an array of strings.) [4 points]
- Implement sort_chunks(). Using your favorite sorting algorithm, sort the array containing the message chunks. Sort based on the first character in the chunk - it will always be a number. We provide a swap_strings function that you may use. Example usage: swap_strings(chunks[0], chunks[1]) will swap the contents of the first and second string. [10 points]

3 Submission

The submission for this assignment has one part: a source code submission. The files should be attached to the homework submission link on BlackBoard.

Writeup: For this assignment, no write up is required.

Source Code: Name your files as “LastNameComputation.c”, “LastNameCollatz.c”, "LastNameDecrypt.c" (e.g. "AcunaComputation.c”, “AcunaCollatz.c”, and “AcunaDecrypt.c").