Computer Science 122 Computer Architecture & Assembly Language

Programming Project 1 – C Programming

Due 1/14/21

This project assumes that you are able to create and run simple C programs using Online GDB or a Visual C++ Solution in Microsoft Visual Studio. If you do not know how to create a Visual C++ Solution in Microsoft Visual Studio, or write a program in Online GDB please see me.

Part I

Please write an individual C for each of the following prompts:

Figure 2.6 on page 63 (1 point)

Figure 2.10 on page 66 (1 point)

With each of the above listed programs, your first task is to simply copy the code from the textbook into Online GDB or a Visual C++ Win32 Console Application project. Get these programs to run as is, with little or no changes to the code listed in the textbook.

When you are sure the programs run correctly, modify each program by adding a **`printf`** statement before each **`scanf`**, to prompt the user as to what input is needed by the program. For example, for the program in Figure 2.10, you must add a prompt asking the user to "Enter an integer value between 0 and 100: " just before the **`scanf`** statement.

Part II

Please create an individual Online GDB or Visual C++ project for each of the following exercises. Remember to place your name, date, project, and course information at the top of each source code file in a header comment:

Problem 8 on page 111 (2 points):

8. Write a C program that inputs two integers and outputs their quotient and remainder. To output the % character, you must write it as %% in the format string.

Sample Input

```
Enter two integer values: 13  4
```

Sample Output

```
13/4 has value 3

13%4 has value 1
```

Problem 59 on page 180 (6 points):

Defining a binary number as in Problem 57, write the function:

```
void decToBin(int bin[], int dec)
```

to convert a nonnegative decimal integer into an eight-bit unsigned binary number. Test your function with interactive input, meaning that your `main()` function should prompt the user to enter the nonnegative decimal integer, invoke your `decToBin()` function, passing a reference to an array and the user-entered nonnegative decimal integer. The `decToBin()` function should convert the nonnegative decimal integer into an eight-bit unsigned binary number, storing each bit in the `bin[]` array, and then display all of the bits in the `bin[]` array using a loop. Here are four separate sample test runs. Like my program, your program should only convert one decimal integer number per run:

Sample test run #1:

```
Enter a nonnegative integer number between 0 and 255: 0
0 converted to unsigned binary is:
0 0 0 0 0 0 0 0


...Program finished with exit code 0
Press ENTER to exit console.
```

Sample test run #2:

```
Enter a nonnegative integer number between 0 and 255: -9
-9 is not a nonnegative integer number.


...Program finished with exit code 0
Press ENTER to exit console.
```

Sample test run #3:

```
Enter a nonnegative integer number between 0 and 255: 1024
1024 is greater than 255 and will require more than eight bits.


...Program finished with exit code 0
Press ENTER to exit console.
```

Sample test run #4:

```
Enter a nonnegative integer number between 0 and 255: 237
237 converted to unsigned binary is:
1 1 1 0 1 1 0 1

...Program finished with exit code 0
Press ENTER to exit console.
```