CS 201: Computer Systems

Assignment 1: C Language and Integer Representation

Academic Integrity

You may NOT, under any circumstances, begin a programming assignment by looking for completed code on StackOverflow or Chegg or any such website, which you can claim as your own. Please check out the Student Code of Conduct at PCC.

The only way to learn to code is to do it yourself. The assignments will be built from examples during the lectures, so ask for clarification during class if something seems confusing. If you start with code from another source and just change the variable names or other content to make it look original, you will receive a zero on the assignment.

I may ask you to explain your assignment verbally. If you cannot satisfactorily explain what your code does, and answer questions about why you wrote it in a particular way, then you should also expect a zero.

Purpose

The purpose of this assignment is to be able to write and compile C programs, be able to use bitwise logic operators, understand integer type casting and be able to convert from one type to the other.

After completing this assignment you will be able to:

- Explain the basic syntax of C, demonstrate C Functions and Command Line Arguments

- Understand Bitwise Operators
- Understand how to represent signed and unsigned integers
- Explain conversion between signed and unsigned integers

Tasks

Part A: Convert C++ to C

- In the assignment folder, there is a file called `payroll.cpp` file. This is written in C++ so it will not compile with a C compiler.
- You will need to create a new file called `payroll.c` and convert the code in payroll.cpp to compile in gcc. Please make sure the output is exactly the same as the output in `payroll.cpp`.
- NOTE: You won't need to change much to get it to compile and run so don't do massive changes to it.
- NOTE: To compile your C code, use gcc NOT g++. Note that most of the options for g++ are basically the same in gcc. So to compile and run, use:

```
gcc payroll.c -o payroll
./payroll
```

- Remember to change the extension on the final C file to .c

Part B: Bitwise Operators

- Download the file called `bitwise.c` .
- In the program there is a function that takes an unsigned integer num (note this is always positive or zero) and an int position pos (0..31) and returns the value of the (pos)th bit as an integer (this will be either 1, 0 or -1 for error). Check for pos being between 0 and 31 and if it's not return a -1.

Criteria for Success

❑ Need to use at least one bitwise logic operator and at least one shift operator in your function somewhere.

Sample Runs

```
----------------------
EXAMPLE RUN #1:
----------------------
enter number: 8
enter position (0 to 31): 3
value is 1
```

```
-----------------------
EXAMPLE RUN #2:
```

```
-----------------------
enter number: 8
enter position (0 to 31): 2
value is 0
```

```
----------------------
EXAMPLE RUN #3:
-----------------------
enter number: 1000
enter position (0 to 31): 100
value is -1
```

Part C: Mixed Comparisons

- Download the `mixed.c` file.
- Compile and run the program with several positive, negative and zero values. Some suggested values are: 20, -20, 200, -200, 0.
- Explain why these values work or don't work in the comments at the top of the file.
- Then change the function to work the way it should. Note that you should get a <= printout if the number is negative.
- NOTE: Change the function ONLY. Don't touch the rest of the program. And don't change the types (however, typecasting is okay). Also, put your explanations inside comments in the file.

Part D: Typecasting

- Download the `typecast.c` file.
- The program inputs into a function an int type and then typecasts this number to various sizes and unsigned types.
- Explain the output for the following values: 1000, -1000, 100, -100. Think about unsigned/signed encoding and overflow in your answer.
- Why do some numbers end up negative when the value was positive? Why do some small numbers end up with a huge magnitude when typecast to unsigned?
- Put your explanation in the comments inside the file.

Criteria for Success for all Parts

- For information on how to do this assignment, see week 1 and week 2 lecture pages in the D2L shell.
- Test your program using any given sample runs or your own data multiple times.
- Upload the four C source code files, with your answers in comments, per the instructions in the D2L Assignment folder for assignment 1 by the due date. ***Some instructors require you to turn in your assignment on D2L while some others like it turned into the Linux server. Be sure to read the instructions in the assignment folder.***
- Do your own work. Consult the syllabus for more information about academic integrity.

Grading Criteria

Part A:

- Compiles with the following command:

```
gcc payroll.c -o payroll
```

- Produces the same output as before converting the code.

Part B:

- Compiles with the following command:

```
gcc bitwise.c -o bitwise
```

- Produces accurate output.
- Only the given function `bit_value` is modified.
- To get any credit, need to use at least one **bitwise logic operator** and at least one **shift operator** in your function somewhere.

Part C:

- Modified code compiles and runs without errors.
- Accurate and detailed description of what is produced and why it is.
- Only the given function `compares` is modified.
- Doesn't change the declaration types on the variables (including parameters). However, you can use a typecast.

Part D:

- Accurate and detailed description of what is produced and why it is.