ECE 209 Program 1: nums

Summer 2018

Due Monday, June 4 @ 11:45pm

This programming assignment must be completed individually. Do not share your code with or receive code from any other student. Evidence of copying or other unauthorized collaboration will be investigated as a potential academic integrity violation. **The minimum penalty for cheating on a programming assignment is a grade of -100 on the assignment.** If you are tempted to copy because you're running late, or don't know what you're doing, you will be better off missing the assignment and taking a zero. Providing your code to someone is cheating, just as much as copying someone else's work.

DO NOT copy code from the Internet, or use programs found online or in textbooks as a "starting point" for your code. Your job is to design and write this program from scratch, on your own. Evidence of using external code from any source will be investigated as a potential academic integrity violation.

This program will convert a base-N number, where N ranges from 2 to 36, to decimal, hexadecimal and binary.

The learning objectives of the program are:

- Write a complete C program.
- Use loops and conditional statements.
- Use printf and scanf to perform I/O.

Program Specification

The entire program must be written in a single source code file, named **nums.c**.

The program will ask the user to the base. This must be a number between 2 and 36 (inclusive). If zero is entered, the program quits. If an illegal base is entered, the program asks again.

The prompt message and the error message must look exactly like the example below. (Characters typed by the user are highlighted in gray. It won't look like this on the console, of course, but the highlighting is done here to emphasize what is entered by the user.)

```
Enter the base (2-36, 0 to quit): 13
```

NOTE: You may assume that the user will type exactly one decimal number, followed by linefeed (Enter) in response to the base prompt. It may be an illegal number, but it will be a number.

Once an acceptable base (we'll call it N) has been entered, the program will ask the user to enter the base-N value to convert. For values 0-9, the decimal digits '0' through '9' will be used. For greater than 9, letters of the alphabet are used: 'A' for 10, 'B' for 11, 'C' for 12, etc., up to 'Z' for 35. Either upper-case or lower-case letters may be used.

Example:

```
Enter the base (2-36, 0 to quit): 13
Enter the base 13 value: 3a7c
```

Note that the base is printed in the prompt message.

If any illegal character is included in the number (e.g., a non-alphanumeric character, or a character that is outside the range of base-N digits), an error message is printed (see appendix) and the user is prompted for the next base.

The program then checks whether the value is less than 65,536 (decimal) -- if it is too large, an error message is printed (see appendix), and the user is prompted for the next base. This restriction on the value allows it to be printed using a 16-bit binary representation. ($65{,}536 = 2^{16}-1$)

The program then prints the decimal, hexadecimal, and binary representations of the value.

- Decimal value has no leading zeroes.
- Hexadecimal value begins with "0x", and it uses lower-cases letters for digits A-F.
- Binary value is exactly 16 bits, including leading zeroes if necessary.

Example:

```
Enter the base (2-36, 0 to quit): 13
Enter the base 13 value: 3a7c
Decimal: 8384
Hexadecimal: 0x20c0
Binary: 0010000011000000
```

After printing, the program prompts for a base again. When the user enters a base of zero, the program quits, returning EXIT_SUCCESS.

Testing the Program

You can use your calculator to confirm that the decimal, hex, and binary values are all the same. Work out some multi-digit weird-base numbers by hand to make sure that you're calculating the values correctly. (See the example run for some test cases, but make up some of your own to make sure.)

Make sure you test all the error conditions, as well. What happens when an illegal base is entered? What happens when an out-of-range digit is part of the number? What happens when an illegal character is part of the number? What happens if the value of the number is larger than $2^{16}-1$?

The output of your program must EXACTLY match the example runs in the Appendix. I mean exactly, down to the spaces and linefeeds. Any deviation will result in deducted points. (We test your program using a script that compares your output text to ours -- any difference will be flagged.)

Hints and Suggestions

Use char constant values instead of ASCII codes

When you are writing the code, try use char literals (e.g., '0') instead of ASCII values (e.g., 0x30 or 48). Your code will be easier to read and understand, because anyone reading the code can see that you're

talking about the character zero. If you use the ASCII value, your reader will either have to remember what character corresponds to that value, or will have to look it up in the ASCII table.

Remember: Sometimes the "reader" of the code is you! When you work on the code over time, you will have to understand what you wrote before. Therefore, it's in your best interest to make the code as easy to read and understand as possible.

Reading in the value

Read the value one character at a time. DO NOT use a string. We have not discussed strings in class yet, and there's no need for it. Even if you know how to use strings, don't use them for this program.

Each character corresponds to a single digit of the value. You must convert the ASCII character to a numerical value (i.e., '0' becomes 0, '1' becomes 1, 'A' becomes 10).

Since we're reading the most significant digit first, we have to "shift" it over an appropriate amount. The problem is that we don't know how much to shift it, because we don't know how many digits will be entered. So here's what we do instead:

1. Initialize value to 0.
2. Read in a new digit; convert from character to numerical value.
3. "Shift" old value to the left, by multiplying by the base (N).
4. Add the new digit to the value.
5. Go back to 2, and repeat until a linefeed character ('\n') is read.

Bogus characters

When you find an illegal character in your value, you can stop trying to calculate the value. However, you must keep reading the characters until you see a linefeed -- otherwise, scanf will try to read these characters as the next base to be entered.

(You'll probably discover this on your own, and there will probably be Piazza questions about it, but I figured I'd put it here anyway, in case someone actually reads it.)

Printing the value

For decimal and hexadecimal, it's easy: printf does the work for you.

There's no printf code to print in binary, though, so you will need to look at each bit in the number and print a '1' or '0' accordingly. We've specified a 16-bit binary representation to make this a bit easier for you.

Other thoughts

- Don't overcomplicate the program. Do not use anything that we have not covered in class. (For example, don't try to use an array or a string.)
- You must include **stdio.h** for **printf/scanf** and **stdlib.h** for EXIT_SUCCESS.
- Write your functions in a general way. In other words, don't write code for all of the specific bases (2, 3, 4, ..., 36). Write a single algorithm that will work for any base.
- However, a good way to think about the problem is to consider the code for a few specific cases, and then generalize that code to handle all cases.

- For compiler errors, look at the source code statement mentioned in the error. Try to figure out what the error is telling you. Try to fix the first error first, and then recompile. Sometimes, fixing the first error will make all the other errors go away. (Because the compiler got confused after the first error.)
- Use a source-level debugger, like *kdbg*, to step through the program if the program behavior is not correct. If you are using NetBeans on your own computer, the debugger is integrated with the editor and compiler, so there's no excuse for not using it.
- For general questions or clarifications, use the Message Board, so that other students can see your question and the answer. For code-specific questions, email your code (as an attachment) to the support list: ece209-sup@wolfware.ncsu.edu.

Administrative Info

Updates or clarifications on Piazza:

Any corrections or clarifications to this program spec will be posted on Piazza. It is important that you read these postings, so that your program will match the updated specification.

What to turn in:

- Source file – it must be named **nums.c**. Submit via Moodle to the Program 1 assignment.

Grading criteria:

20 points: File submitted with the proper name. (This is all or nothing. Either you will get all 20 points, or you will get zero. Be sure your file is named correctly, and remember that case matters.)

20 points: Compiles with no warnings and no errors (using –Wall and –pedantic). (If the program is not complete, then only partial credit is available here. In other words, you won't get 20 points for compiling a few trivial lines of code.)

10 points: Proper coding style, comments, and headers. No unnecessary global variables. No goto. (See the Programs web page for more information.)

10 points: Base and value are read correctly, assuming no illegal inputs.

10 points: Value is correctly converted from base-N.

10 points: Decimal and hexadecimal representations are printed correctly.

10 points: Binary representation is printed correctly.

5 points: All illegal inputs are handled correctly.

5 points: Program flow is correct -- keep asking for conversions until base 0 is entered.

Appendix: Example Runs

Text entered by user is highlighted.

Case 1: All legal inputs

Note that base is printed as part of the prompt for the value.

```
Enter base (2-36, 0 to quit): 5
Enter base 5 value: 1234
Decimal: 194
Hexadecimal: 0xc2
Binary: 0000000011000010
Enter base (2-36, 0 to quit): 36
Enter base 36 value: xyz
Decimal: 44027
Hexadecimal: 0xabfb
Binary: 1010101111111011
Enter base (2-36, 0 to quit): 2
Enter base 2 value: 10100001110
Decimal: 1294
Hexadecimal: 0x50e
Binary: 0000010100001110
Enter base (2-36, 0 to quit): 0
```

Case 2: Illegal base

```
Enter base (2-36, 0 to quit): 1
Enter base (2-36, 0 to quit): 40
Enter base (2-36, 0 to quit): -1
Enter base (2-36, 0 to quit): 0
```

Case 3: Illegal values

Note that the illegal character is printed as part of the error message. If there are multiple illegal characters, only the first one is printed. ('*' is illegal because it's not alphanumeric; 'G' is illegal because it does not represent a base-16 digit.) In the last case, the character 'Z' ('z') is legal, but the number is larger than 16 bits.

```
Enter base (2-36, 0 to quit): 10
Enter base 10 value: 12*15
Bogus character * -- value ignored.
Enter base (2-36, 0 to quit): 16
Enter base 16 value: 01abGF
Bogus character G -- value ignored.
Enter base (2-36, 0 to quit): 36
Enter base 36 value: ZzZzZzZz
ERROR: value out of range
Enter base (2-36, 0 to quit): 0
```