To Do’s for Lab Week 3

CHRIS NIELSEN

Part 1

EXERCISE 1 PART1)

c. Use printf %x, %d, and %c to display the results. Explain the differences.

Exercise 3:

Follow the next step and record and <u>submit in a separate text file the items in italics in addition to the C code for the exercise</u>.

1) What steps would solve the problem?

   Describe in common language the solving of the exercise.

   The steps that would solve the problem go like this-
   Input tne variable of the hexadecimal, then | the mask with it so then regardless of what the numbers are ast the least 4 significant digits you always get an F at the last byte

2) Identify the variables that are needed to solve the exercise.

   Indicate the needed variables and specify their purpose.

   Unsigned int, unsigned int, both of these will hold the input from the user and the mask (0x0f) respectively.

3) Identify the type of the variables.

   Indicate the type for each variable and explain your choice.

   The types i chose were unsigned ints because they work much better than char since using scans f on a char always results in undefined behavior.

4) Read the input information from the keyword and assign the information to the related variables.

   Describe your understanding of the used C instructions.

   My understanding is that the printing scanning and | bitwise operators all come together to find the corect value and i used them in an efficient way.

5) Display the values of each variable.

   Explain if the displayed value was correct. If it was not, what change did you do to your program, and why?

It was correct, it was a little confusing at first but i got it!

6) What bitwise operation is needed? What mask is necessary?

   Explain your reasoning.

   |, that way we can preserve the correct values for the first 4 digits and stay true to the prompt. Mask that is necessary is (0x0f)

7) Display the required output.

   Describe the output. If it is not, what error did you get, and how do you plan to fix it? Did your correction solve the problem?

   The output was appropriate for the inputs that I tested it with.

How similar was the solution at Step 1 to the final solution?

Very similar since I wrote this after the fact whoops.

Part 2

Exercise 1:

Follow the next step and record and <u>submit in a separate text file the items in italics in addition to the C code for the exercise</u>.

1) What steps would solve the problem?

   Describe in common language the solving of the exercise.

   The steps that would solve the problem go like this-
   Define an unsigned char , 2 ints
   Read the appropriate values from the user, then generate the mask by moving first left(7-p) and that limits the least significant bits and then you move back over in order to position it correctly with (8-n). Then we print it !

2) Identify the variables that are needed to solve the exercise.

   Indicate the needed variables and specify their purpose.

   Unsigned char, 2 ints
   You need them to hold the p and n inputs respectively and then the input by the user.

3) Identify the type of the variables.

   Indicate the type for each variable and explain your choice.

   Unsigned char is told to us to use and since the values were Ovid by are integers we need integer value placeholders.

4) Read the input information from the keyword and assign the information to the related variables.

   Describe your understanding of the used C instructions.

We scan like we always have, and then due to the appropriate positioning we use 7 and 8 respectively since we start all the way at the front of the bit strings no then need to position it within the 8 bit span.

5) Display the values of each variable.

    Explain if the displayed value was correct. If it was not, what change did you do to your program, and why?

    It was correct thank goodness!

6) What shift operation is needed to achieve the required bit positioning? How are the remaining bits set to 0?

    Explain your reasoning.

    The remaining bits are set to zero by moving over to the right and having those 0 bit values file in

7) Display the required output.

    Describe the output. If it is not, what error did you get, and how do you plan to fix it? Did your correction solve the problem?

    Yes, it was correct

How similar was the solution at Step 1 to the final solution?

The same.

EXCERSISE 2 PART 2

2) Repeat exercise 1 using variables of type unsigned int instead of unsigned char. Discuss

the differences between the two implementations.

The differences between the two variable types include the ways they interact with SCANF and the undefined behavior that comes to the amount of info they can hold respectively, i never understood hoe to FORCE unsigned char to work sometimes it works??? The structure i used is the same i just swapped the values and can't find anything online to help with that.

Exercise 3:

Follow the next step and record and <u>submit in a separate text file the items in italics in addition to the C code for the exercise</u>.

1) What steps would solve the problem?

    Describe in common language the solving of the exercise.

    The steps that would solve the problem go like this-

    Read all the hexadecimals, P and N, then you should generate the seperate masks and deal with the two variables separately. Generate the first mask to save the correct values at the correct position for one, at which point you negate it and AND it it's the mask. This will clear those spaces, always making them 0. Then for two we have to get

the values from two at the correct positions and replace it so we go ahead and get the correct amount of integers we want from var 2 bshoftong and them masking, then finally we or it transfers the data.

Identify the variables that are needed to solve the exercise.
Indicate the needed variables and specify their purpose.
Unsigned ints, and ints for the mask and hexadecimals respectively

2) Identify the type of the variables.
   Indicate the type for each variable and explain your choice.
   Unsigned char is told to us to use and since the values were Ovid by are integers we need integer value placeholders.

3) Read the input information from the keyword and assign the information to the related variables.
   Describe your understanding of the used C instructions.
   They all shift and use bit wise operations in order to transfer the values correctly
4) Display the values of each variable.
   Explain if the displayed value was correct. If it was not, what change did you do to your program, and why?
   It was right according to the test cases!
5) What bit level operations are needed to achieve the required bit replacement?
   Explain your reasoning.
   You need ~ | and &
6) Display the required output.
   Describe the output. If it is not, what error did you get, and how do you plan to fix it? Did your correction solve the problem?

   The desired output was what the test cases said.

How similar was the solution at Step 1 to the final solution?

Different cause it took a while to figure out

Part 3

Exercise 1

Follow the next step and record and <u>submit in a separate text file the items in italics in addition to the C code for the exercise</u>.

1) How is this exercise like the pocket calculator exercise discussed in class? What is the same and what is different?
   Describe in common language the solving of the exercise, and how it relates to the pocket calculator exercise discussed in class.

The steps that would solve the problem go like this-

How it's similar is,

2) Identify the variables required in the program.
Indicate the needed variables and specify their purpose. Indicate the reasons for the selected variable types.

Unsigned char because it tells us to, ints for the results cause it works better than char, ints for the masks, and ints for the p and n and the char for the string in the switch case

3) Read the needed values from the keyboard.
Describe your understanding of the used C instructions. How do you check it the values were read correctly?

The values were read correctly because I tested them every way they could be inoutted and due to that I was able to see their always correct outputs.

4) Read a character representing the command.
How is this read instruction different from the previous read instruction? How do you check it the value was read correctly?

You can check if it defaults and doesn't run anything since the command runs off of reading a string properly

5) What bitwise operations are needed for each commend? What masks are used with each bitwise operation? **You can't actually use a single mask and just negate it. You need | & and ^**

Explain your reasoning. Cause you need to replace or swap or negate certain values

6) Perform the bitwise operation corresponding to the operation.
7) Display the result.
Explain if the displayed value was correct. If it was not, what change did you do to your program, and why?

It wasn't correct most of the it but then n i fixed it with the mask generation and the correct bit wise operations now it works under the test cases

8) Modify your code so that it can repeatedly execute steps 3-6 for other values and another command character.
Explain in common language your solution for the repetitive actions.
Explain the actions that you expect from each C instruction to achieve the repetitions. You can just just a while true loop
Explain if the displayed values were correct. If it was not, what change did you do to your program, and why?

You would make a choice to enter a while loop which would have all the same code and never exit. That would be another CASE in the switch. This would be great to section off. The case in the case could be L for loop.

How different was your end program from the solution that you created at Step 1? It was pretty different as i had no idea how to make it in the beginning.

EXCERSIZE 3 PART 2

2) Repeat exercise 1 using a variable of type unsigned int. Discuss the differences between

the two programs.

The differences between the two variable types include the ways they interact with SCANF and the undefined behavior that comes to the amount of info they can hold respectively, i never understood hoe to FORCE unsigned char to work sometimes it works??? The structure i used is the same i just swapped the values and can't find anything online to help with that. This version using just unsigned int was so much better