**Lab Activity 2**

**ESE 124 (Programming Fundamentals)**

**Developed by Dr. Alex Doboli**

**Extended and edited by Mrs. Jenny Chen**

**Note***: You will earn 30% of your lab grade for completing and submitting your lab work at the end of the lab activity. You must submit the code that you developed for the exercises that you worked on during the lab, and the three associated to-dos. The TODOs should be short. Please don’t spend more than 3 minutes on each. You will be graded on your effort during the lab session. You will not be graded on the correctness of the submission.*

*The remaining 70% will be earned after submitting your final solutions for the lab exercises, and they will be graded on correctness to the test cases.*

*For both submissions, each lab exercise is 100 points. The grade for the lab is the average number of points for the submitted exercises.*

***The homework exercises*** *are submitted together with the lab exercises, the latest before the start of your next week’s lab session.* *Each homework exercise is 100 points. The grade for the homework is the average number of points for the submitted exercises.*

Goal

This lab activity focuses on the four following topics: **ASCII tables, bit-level operators, format specifiers in C, and writing a C program to manipulate bitwise operators.**

## Submission

Please follow the instructions posted on Brightspace. Your final solutions are due prior to the start of your next week’s lab session.

# Lab Activities

## **Part 1** **[ 1 Hour ]**

**Description**

In this exercise, you will find the byte sizes of the different data types, and use format specifiers to input and output with *scanf()* and *printf()* functions. Function *printf()* is used to display the output and function *scanf()* is used to read the inputs from the keyboard. Use operator *sizeof ()* to find the byte size (e.g., number of bytes) of each of the above types.

**Exercise 1**: Bit-level operations use variables of the following types:

* unsigned char
* unsigned short
* unsigned int
* unsigned long

1. Use sizeof operator to find the byte size (e.g., number of bytes) of each of the above types.
2. Assign the following hexadecimal constants to four C variables of type unsigned char – 0x12; 0xda; 0x3b; 0xbe.
   1. Initialize the variables when you define them
   2. Use scanf %x
   3. Use printf %x, %d, and %c to display the results. Explain the differences.
   4. Perform the following bitwise operations between the first variable and the second variable; and between the third and fourth variables: bitwise and, bitwise or, and bitwise exclusive or. Display the results using printf %x and manually verify the results.
3. Write a C program that reads one hexadecimal number from the keyboard (using scanf %x) and then sets the least significant 4 bits of the number to 1111. The resulting value is displayed using printf %x.

**Note**: The following test cases are not used in grading but to clarify the requirements of exercise 3.

**Test case 1:**

**Input:**

Enter a hexadecimal number: 0x5A

**Output:**

Result: 0x5F

**Test case 2:**

**Input:**

Enter a hexadecimal number: 0x0

**Output:**

Result: 0xF

Submit C code for **Exercise 1 and the related TODO file.**

**Part 2 [ 1 Hour ]**

**Description**

In this exercise, you will learn about **bit manipulations**. You can read more at the following link [more](https://codeforwin.org/c-programming/bitwise-operator-programming-exercises-and-solutions-in-c):

**Exercise 1**: Write a C program that reads a hexadecimal value from the keyboard and then stores the value into an unsigned char variable. Read two int values p and n from the keyboard, where the two values are less than 8. Change the initial hexadecimal value in the following way: Shift the n bits starting at position p, so that they form the n least significant bits of the result. The remaining bits of the result are set to 0. Display the result using printf %x.

**Note**: The following test cases are not used in grading but to clarify the requirements of exercise 1.

**Test Case 1:**

**Input:**

Enter a Hexadecimal value (0x00-0xff): 0x12

Enter p and n (p,n): 5 2

**Output:**

0x01

**Test Case 2:**

**Input:**

Enter a Hexadecimal value (0x00-0xff): 0x1f

Enter p and n (p,n): 4 3

**Output:**

0x07

**Exercise 2**: Repeat exercise 1 using variables of type unsigned int instead of unsigned char. Discuss the differences between the two implementations.

**Note**: The following test cases are not used in grading but to clarify the requirements of exercise 2.

**Test Case 1:**

**Input:**

Enter a Hexadecimal value (0x00-0xff): 0x12

Enter p and n (p,n): 5 2

**Output:**

0x01

**Test Case 2:**

**Input:**

Enter a Hexadecimal value (0x00-0xff): 0x1f

Enter p and n (p,n): 4 3

**Output:**

0x07

**Exercise 3**: Write a C program that reads two hexadecimal values from the keyboard and then stores the two values into two variables of type unsigned char. Read two int values p and n from the keyboard, where the values are less than 8. Replace the n bits of the first variable starting at position p with the last n bits of the second variable. The rest of the bits of the first variable remain unchanged. Display the resulting value of the first variable using printf %x.

**Note**: The following test cases are not used in grading but to clarify the requirements of exercise 3.

**Test Case 1:**

**Input:**

Enter the first hexadecimal value: 0x1f

Enter the second hexadecimal value: 0xc3

Enter p and n(p,n):4 3

**Output:** 0x0f

**Test Case 2:**

**Input:**

Enter the first hexadecimal value: 0x1f

Enter the second hexadecimal value: 0xc3

Enter p and n(p,n):5 2

**Output:** 0x3f

Submit the C code for **exercises 1, 2 and 3.** Submit **one TODO** file that summarizes what you learned in the three exercises.

**Part 3 [ 50 minutes ]**

**Exercise 1**: Develop a C program that reads a hexadecimal value from the keyboard and then stores the value into an unsigned char variable. Read two int values p and n from the keyboard, where the values are less than 8. Implement the following commands:

S – sets the n bits starting at position p to 11..1

R – resets the n bits starting at position p to 00…0

F – flips the n bits starting at position p to their inverse bit

D – displays the value of the variable

I – reads a new hexadecimal value from the keyboard

**Note:** The following examples clarify the functionality of the program. The examples are not for grading.

Perform the operations S, R, F, D, I on the outputs you have generated previously as follows:

Input->Output1->Output2-> Output3->..

You should not perform operations only on the initial input as the following:

Input->Output1  
Input->Output2  
Input->Output3

**Test case 1:**

Enter a Hexadecimal input: 0x12

Enter p and n(p,n): 5,2

Please enter an operation 'S', 'R', 'F', 'D', 'I': S

Output: 0x32

Please enter an operation 'S', 'R', 'F', 'D', 'I': R

Output: 0x02

Please enter an operation 'S', 'R', 'F', 'D', 'I': F

Output: 0x32

Please enter an operation 'S', 'R', 'F', 'D', 'I': F

Output: 0x02

Please enter an operation 'S', 'R', 'F', 'D', 'I': I

Enter a Hexadecimal input: 0x10

Please enter an operation 'S', 'R', 'F', 'D', 'I': D

Value of the input is 16

Please enter an operation 'S', 'R', 'F', 'D', 'I':

**Test case 2:**

Enter a Hexadecimal input: 0x6

Enter p and n(p,n): 2,2

Please enter an operation 'S', 'R', 'F', 'D', 'I': F

Output: 0x00

Please enter an operation 'S', 'R', 'F', 'D', 'I': F

Output: 0x06

Please enter an operation 'S', 'R', 'F', 'D', 'I': R

Output: 0x00

Please enter an operation 'S', 'R', 'F', 'D', 'I': S

Output: 0x06

Please enter an operation 'S', 'R', 'F', 'D', 'I': D

Value of the input is 6

Please enter an operation 'S', 'R', 'F', 'D', 'I': I

Enter a Hexadecimal input: 0xa7

Please enter an operation 'S', 'R', 'F', 'D', 'I': F

Output: 0xa1

Please enter an operation 'S', 'R', 'F', 'D', 'I': F

Output: 0xa7

Please enter an operation 'S', 'R', 'F', 'D', 'I':

**Exercise 2**: Repeat exercise 1 using a variable of type unsigned int. Discuss the differences between the two programs.

**Test case 1:**

Enter a Hexadecimal input: 0x12

Enter p and n(p,n): 5,2

Please enter an operation 'S', 'R', 'F', 'D', 'I': S

Output: 0x32

Please enter an operation 'S', 'R', 'F', 'D', 'I': R

Output: 0x02

Please enter an operation 'S', 'R', 'F', 'D', 'I': F

Output: 0x32

Please enter an operation 'S', 'R', 'F', 'D', 'I': F

Output: 0x02

Please enter an operation 'S', 'R', 'F', 'D', 'I': I

Enter a Hexadecimal input: 0x10

Please enter an operation 'S', 'R', 'F', 'D', 'I': D

Value of the input is 16

Please enter an operation 'S', 'R', 'F', 'D', 'I':

**Test case 2:**

Enter a Hexadecimal input: 0x06

Enter p and n(p,n): 2,2

Please enter an operation 'S', 'R', 'F', 'D', 'I': F

Output: 0x00

Please enter an operation 'S', 'R', 'F', 'D', 'I': F

Output: 0x06

Please enter an operation 'S', 'R', 'F', 'D', 'I': R

Output: 0x00

Please enter an operation 'S', 'R', 'F', 'D', 'I': S

Output: 0x06

Please enter an operation 'S', 'R', 'F', 'D', 'I': D

Value of the input is 6

Please enter an operation 'S', 'R', 'F', 'D', 'I': I

Enter a Hexadecimal input: 0xa7

Please enter an operation 'S', 'R', 'F', 'D', 'I': F

Output: 0xa1

Please enter an operation 'S', 'R', 'F', 'D', 'I': F

Output: 0xa7

Please enter an operation 'S', 'R', 'F', 'D', 'I':

***Submit the C code for exercises 1 and 2. Submit one TODO file for both exercises to summarize what you learned through the two exercises.***

# **Homework**

## **Exercise 1**: Write a C program that reads one hexadecimal number from the keyboard (using scanf %x) and then sets the most significant 2 bits of the number to 00 and the least significant 2 bits to 11. The resulting value is displayed using printf %x.

**Note:** The following examples clarify the functionality of the program. The examples are not for grading.

**Test case 1:**

**Input:**

Enter a hexadecimal number: 0x12

**Output:**

0x13

**Test case 2:**

**Input:**

Enter a hexadecimal number: 0x5A

**Output:**

0x1b

**Exercise 2**: Write a C program that reads one hexadecimal number from the keyboard (using scanf %x) and then sets the most significant p bits of the number to 0 and the least significant 8 - p bits to 1. The resulting value is displayed using printf %x. The value of variable p is read from the keyboard and is between 0 and 8.

**Note:** The following examples clarify the functionality of the program. The examples are not for grading.

**Test case 1:**

**Input:**

Enter a hexadecimal number: 0x15

Enter a value for P: 4

**Output:**

f

**Test case 2:**

**Input:**

Enter a hexadecimal number: 0x3A

Enter a value for P: 6

**Output:**

3

**Exercise 3**: Devise: a C program that counts the number of bits 1 and separately the number of bits 0 in a binary number of length 4 bytes.

**Note:** The following examples clarify the functionality of the program. The examples are not for grading.

**Test case 1:**

**Input:**

Enter a 32 bit Hexadecimal value: 0xFFFFFFFF

**Output:**

Number of ones: 32

Number of zeroes: 0

**Test case 2:**

**Input:**

Enter a 32 bit Hexadecimal value: 0x55555555

**Output:**

Number of ones: 16

Number of zeroes: 16

## **Exercise 4**: Write a C program that counts the number of leading bits 0 in a binary number of length 4 bytes.

**Note:** The following examples clarify the functionality of the program. The examples are not for grading.

**Test case 1:**

**Input:**

Please enter a 32 bit hexadecimal number:0x0000FFFF

**Output:**

Number of leading zeros are 16

**Test case 2:**

**Input:**

Please enter a 32 bit hexadecimal number:0x12345678

**Output:**

Number of leading zeros are 3