ENGG*1410: "Introductory Programming for Engineers", Assignment #11 "Operations on Bits"

Prof. Shawki Areibi
School of Engineering, University of Guelph
Fall 2021

Start Date: Week #11, Due Date: Week #12 (Friday, 5:00 PM) in Dropbox

You can use the following "Decimal to Binary Converter Utility" to check your answers
Decimal To Binary Converter

1. Write a C program to input any number from user and check whether n^th bit of the given number is set (1) or not (1). For example if the user enters the number "12" and asks if the n^th bit = 2 is set or not, the system will respond with the following message Bit 2 of Number "'12" is set (1). Here is a step by step descriptive logic to get n^th bit of a number
   (a) Input number from user. Store it in some variable say *num*.
   (b) Input the bit position from user. Store it in some variable say *n*.
   (c) To get the n^th bit of *num* right shift *num*, *n* times. Then perform bitwise AND with *1*
2. Write a C program to input any number from the user and count number of trailing zeros in the given number using bitwise operator. For example if the number is $(4)_{10}$ which is equivalent to $(...0000100)_2$ then the number of trailing zero is 2 and if the number is $(48)_{10}$ which is equivalent to $(...0000110000)_2$ the number of trailing zeros is 4.
3. Write a C program to input a number from the user and count total number of ones (1) and zeros (0) in the given number using bitwise operator. For example if the number is $(22)_{10}$ which is equivalent to $(..000010110)_2$ then the system will respond: (a) Number of ones: 3, (b) Number of zeros: 29
4. Write a function called *bitpat_search()* that looks for the occurance of a specified pattern of bits inside an *unsigned int*. The function should take three arguments and should be called as shown:
   bitpat_search (source, pattern, n)
   The function searches the integer *source*, starting at the lefmost bit, to see if the rightmost n bits of *pattern* occur in *source*. If the pattern is found, have the function return the number of bits at which the pattern begins, where the leftmost bit s bit number 0. If the pattern is not found, then have the function reurn -1. So, for example the call
   index = bitpat_search(0xe1f4, 0x5,3);
   causes the *bitpat_search()* function to search the number 0xe1f4 (=1110 0001 1111 0100 binary) for the occurence of the three-bit pattern 0x5 (= 101 binary). The function return 11 to indicate that the pattern was found in the source beginning with bit number 11. Make certain that the function makes no assumptions about the size of an int.