Topics: Bitwise Operators

Bitwise Operators

1. Write a function to print an integer in binary.
2. Write a function to n times right rotate an integer x (rightRot(x, n)).Using this function write a program that takes an integer x and an integer n and outputs x right rotated by n bit positions. Your program must output x in binary and then the rotated x in binary.
3. Even Parity: A bit pattern has even parity if the number of set bits in the pattern is even. (example: 1010, 010111 have even parity, but 1101 does not). Write a function that detects whether an integer has even parity or not.
4. Print (in decimal) the maximum value that can be held in integer
5. Print (in decimal) the maximum value that can be held in unsigned integer
6. Write a function to n times left rotate an integer x (leftRot(x, n)).Using this function write a program that takes an integer x and an integer n and outputs x left rotated by n bit positions. Your program must output x in binary and then the left rotated x in binary.
7. Encoding / Decoding: Encode the following information in 32 bit integer:

    a. Roll number – left most 21 bits

    b. Subject – next 3 bits

    c. Score – next 7 bits

    d. Parity bit – LSB

    Sample subject code:

    - CSE105 – 000
    - CSE106 – 001
    - CSE100S – 010

    Given (a) (b) (c) (d), you should be able to produce the encoded integer. Also given encoded integer, you should be able to decode the information (the score of a particular

student in a particular number). Also, you should be able to validate whether the information got corrupted or not (parity check).

8. Check the IEEE 754 single-precision binary floating-point format from here: https://en.wikipedia.org/wiki/Single-precision_floating-point_format Now, implement this logic to compute a floating point value from the bit pattern of an input integer variable. To check that your calculations are correct, compare your output with the output of the following code, assuming your integer input is stored in a variable called x

```
printf("%f", *((float*)&x))
```