CPSC 121: Models of Computation
Assignment #2

Due: Friday October 10, 5:00 pm **Total: 33 Marks**

Submission Instructions-- read carefully

All assignments should be done in groups of 2. It is very important to work with another student and exchange ideas. **Each group should submit ONE assignment**. Type or write your assignment on clean sheets of paper with question numbers prominently labelled. Answers that are difficult to read or locate may lose marks. We recommend working problems on a draft copy then writing a separate final copy to submit.

Your submission must be **STAPLED** and include the **CPSC 121 assignment cover page** – located at the Assignments section of the course web page. Additionally, include your names at the top of each page. We are not responsible for lost pages from unstapled submissions.

Submit your assignment to the appropriately marked box in room **ICCS X235** by the due date and time listed above. **Late submissions are not accepted**.

Note: the number of marks allocated to a question appears in square brackets after the question number.

A Note on the Marking Scheme

Most items (i.e., question or, for questions divided into parts, part of a question) will be worth 3 marks with the following general marking scheme:

- **3 marks:** correct, complete, legible solution.
- **2 marks**: legible solution contains some errors or is not quite complete but shows a clear grasp of how the concepts and techniques required apply to this problem.
- **1 mark**: legible solution contains errors or is not complete but shows a clear grasp of the concepts and techniques required, although not their application to this problem or the solution is somewhat difficult to read but otherwise correct.
- **0 marks**: the solution contains substantial errors, is far from complete, does not show a clear grasp of the concepts and techniques required, or is illegible.

This marking scheme reflects our intent for you to learn the key concepts and techniques underlying computation, determine where they apply, and apply them correctly to interesting problems. It also reflects a practical fact: we have insufficient time to decipher illegible answers. At the instructor's discretion, some items may be marked on a different scale. TAs may very occasionally award a bonus mark for exceptional answers.

Question 1 [6]

a) Translate the following to (a) unsigned binary notation, and (b) hexadecimal notation. In each case, use the minimum number of digits necessary to represent the value. You MUST show your work to get marks for each part of this question. Just showing the final value is not worth any marks.
   i. 93
   ii. 564
   iii. 4096

b) Consider the following binary numbers. For each number, translate it to a decimal number (a) first assuming it is unsigned, then (b) assuming it is a signed 8-bit number.
   i. 10011010
   ii. 11000111
   iii. 101010

Question 2 [9]

Most programming languages provide operations, called **bit-wise operations** that allow the user to manipulate the bits of signed or unsigned integers. The following list shows some of the bit operations which are available in Java and C. The variables x, y and i represent integers (signed or unsigned):

- **~x** denotes the integer whose i-th bit is the **complement** of the i-th bit of x. That is, the i-th bit of ~x is 0 if the i-th bit of x is 1 and it is 1 otherwise. For instance, assuming that we have 8-bit integers, then $51_{10} = 00110011_2$, and ~51 = 11001100 .
- **x & y** denotes the integer whose i-th bit is the **logical AND** of the i-th bit of x and the i-th bit of y. For instance, assuming that we have 8-bit integers, $109_{10} = 01101101_2$ and $51_{10} = 00110011_2$, then 109 & 51 = $00100001_2 = 33_{10}$.
- **x | y** denotes the integer whose i-th bit is the **logical OR** of the i-th bit of x and the i-th bit of y. For instance, 109 | 51 = $01111111_2 = 127_{10}$.
- **x ^ y** denotes the integer whose i-th bit is the **logical XOR** of the i-th bit of x and the i-th bit of y. For instance, 109 ^ 51 = $01011110_2 = 94_{10}$.
- **x << i** denotes the integer obtained by "deleting" the leftmost i bits of x, and adding i 0 bits to the right of the remaining bits (i.e. it shifts the bits of the number to the left i times and fills in the empty spaces on the right with 0's). For instance, using 8-bit integers,
  since $109_{10} = 01101101_2$, then
  109 << 3 = 01101000 = 104_{10} (the first 3 bits 011 were removed and 3 0s added at the end).
- **x >>> i** denotes the integer obtained by "deleting" the rightmost i bits of x, and adding i 0 bits to the left of the remaining bits (i.e. it shifts the bits of the number to the right i times and fills in the empty spaces on the left with 0's). For instance, using 8-bit integers,
  since $109_{10} = 01101101_2$, then
  109 >>> 3 = 00001101 = 13_{10} (the last 3 bits 101 were removed and 3 0s added at the front).

We can perform several operations by manipulating the sequence of bits that represents an integer using these operators. In this problem, we look at some of them (if you need to, assume that integers are represented by 32 bits using two's complement).

a) There are several situations where we would like to retain only some of the bits of an integer x. For instance, the last 3 bits of 109 equal $101_2 = 5_{10}$. Given that x is an integer, white an expression that would give an integer that corresponds to the **last 4 bits of x**.

b) Now, assuming that x is an integer, white an expression that would give an integer that corresponds to the **next 4 bits of x** , i.e. the bits that correspond to the positions with values 2^4, 2^5, 2^6 and 2^7.

c) If **x** is any integer, what is the result of the following operations?

   i. **x ^ x**

   ii. **x | x**

   iii. **x & x**

   iv. **x << 2** assuming that x is unsigned and the result fits in 32 bits

   v. **x >>> 2** assuming that x is unsigned

   vi. **~x + 1** assuming that x is signed,

<u>Question 3 [6]</u>

Using any necessary rules of inference and equivalences, derive the conclusion from the premises. For each step, indicate on the right the inference rule or equivalence which has been used in the step:

a) $k \rightarrow n$
   $k \vee m$
   $\sim n \wedge q$

   $\therefore\ m \vee s$

b) $(p \vee \sim q) \rightarrow r$
   $r \rightarrow (s \vee t)$
   $\sim s \wedge \sim u$
   $\sim u \rightarrow \sim t$

   $\therefore \sim p$

Question 4 [6]

Using any necessary rules of inference and equivalences, derive the conclusion from the premises. For each step, indicate on the right the inference rule or equivalence which has been used in the step:

a) $p \rightarrow q$
$\sim r \vee t$
$p \vee r$

$\therefore \sim q \rightarrow t$

b) $\sim p \wedge q$
$r \rightarrow p$
$\sim r \rightarrow (s \wedge t)$
$s \rightarrow (t \vee p)$

$\therefore t$

Question 5 [6]

Write the following arguments in symbolic form. Then establish the validity of the argument by providing a proof, or give a counterexample to show that it is invalid:

(a) If Rochelle gets the supervisor’s position and works hard, then she’ll get a raise. If she gets the raise, then she’ll buy a new car. She has not purchased a new car. Therefore either Rochelle did not get the supervisor’s position or she did not work hard.

(b) If there is a chance of rain or she is not wearing her red headband , Lois will not mow her lawn. Unless the temperature is less than 28 degrees, there is no chance of rain. Today the temperature is 30 degrees and Lois is wearing her red headband. Therefore (sometime today) Lois will mow her lawn.