

**UNCA CSCI 235**  
**Exam 2 Fall 2018**  
27 November 2018

This is a closed book and closed notes exam. Communication with anyone other than the instructor is not allowed during the exam. **Furthermore, calculators, cell phones, and any other electronic or communication devices may not be used during this exam.** Anyone needing a break during the exam must leave their exam with the instructor. Cell phones or computers may not be used during breaks.

*This exam must be turned in before 6:55 PM.*

Name: \_\_\_\_\_

**Problem 1 (10 points) C expressions**

In the left column, there are ten tricky and not-so tricky C expressions. Write their values in the right column. Express your answers as simple base 10 expressions, such as 235 or -235. You may assume that all of these numbers are stored in 16-bit two's complement representation, the usual short.

<b>0x64</b>	
<b>050</b>	
<b>19 &amp;&amp; 0</b>	
<b>19 &amp; 27</b>	
<b>19 ^ 27</b>	
<b>37 * 2 / 10</b>	
<b>19 &gt;= 4</b>	
<b>19 &gt;&gt; 4</b>	
<b>19   4</b>	
<b>(7 != 5) * 235</b>	

**Problem 2 (4 points) Q4.4 to decimal conversion**

Convert the following two Q4.4 *two's complement* numbers (four fixed and four fractional bits) into conventional decimal numbers.

<b>10011001</b>	<b>00011000</b>
-----------------	-----------------

**Problem 3 (4 points) Decimal to Q4.4 conversion**

Convert the following two signed decimal numbers into Q4.4 *two's complement* numbers (**four** fixed and **four** fractional bits). If you can't express the number exactly, give the nearest Q4.4 representation.

- 3 . 5	2 . 71828
---------	-----------

**Problem 4 (10 points) Adding numbers with flags**

Add the following pairs of six-bit numbers. Based on the result of this addition, set the four x86-64 status bits: CF (carry), OF (overflow), SF (sign) and ZF (zero).

$\begin{array}{r} 111011 \\ + 000101 \\ \hline \end{array}$	$\begin{array}{r} 010110 \\ + 000110 \\ \hline \end{array}$
CF __, OF __, SF __, ZF __	CF __, OF __, SF __, ZF __
$\begin{array}{r} 101111 \\ + 100000 \\ \hline \end{array}$	$\begin{array}{r} 011000 \\ + 001000 \\ \hline \end{array}$
CF __, OF __, SF __, ZF __	CF __, OF __, SF __, ZF __

**Problem 5 (4 points) Range**

What is the range of numbers that can be stored in 8-bit twos-complement numbers? (The byte of Java is an 8-bit twos-complement number.)

**Problem 6 (4 point) Range**

What is the range of numbers that can be stored in 16-bit unsigned numbers? (The char of Java is a 16-bit unsigned number.)

**Problem 7 (6 points) CSCI arithmetic**

Perform the following operations and express the results as they should be for CSCI 235 and other geeky environments. *Use powers of 2!*

$$16 \text{ ki} * 64 \text{ ki}$$

$$32 \text{ ki} / 128$$

$$\log_2(32 \text{ Gi})$$

**Problem 8 (6 points) Compile and run**

Suppose you have written the following program and stored it in a file named `prob8.c`.

```
#include <stdio.h>
int main(int argc, char *argv[]) {
    printf("%s is %d\n", argv[2], argc) ;
    return 0 ;
}
```

What would be an appropriate **Makefile** for the program? Assume you want to store the executable in the file **prob8**. I am giving you the first line of the **Makefile**. You need to put in the others. Write your **Makefile** so that typing the command **make** (with no arguments) will compile your program.

```
CFLAGS = -std=c99 -Wpedantic -Og -g
```

Now comes the tricky part of the riddle. Below is the start of a single command typed from the command prompt. What command line arguments do you pass to **prob8** to make it print this single output line?

**IV is 4**

Just complete the command.

```
./prob8
```

### Problem 9 (20 points) C Programming

Write a program that reads (**using scanf**) a bunch of highway “numbers” (a string followed by a number) from a terminated standard input stream that are entered as shown below.

```
        US 25  NC
694      I 26  NC 251
```

Your output should be a neatly formatted list of highways (one per line as shown below) followed by a count of the number of NC highways. So, for the above example, the output should something like

```
US      25
NC      694
I       26
NC      251
NC count: 2
```

```
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[]) {

}
```

**Problem 10 (10 points) Boolean expression to truth table and circuit**

**First**, fill in the truth table on the right below so that it corresponds to the following Java (and C) expression:

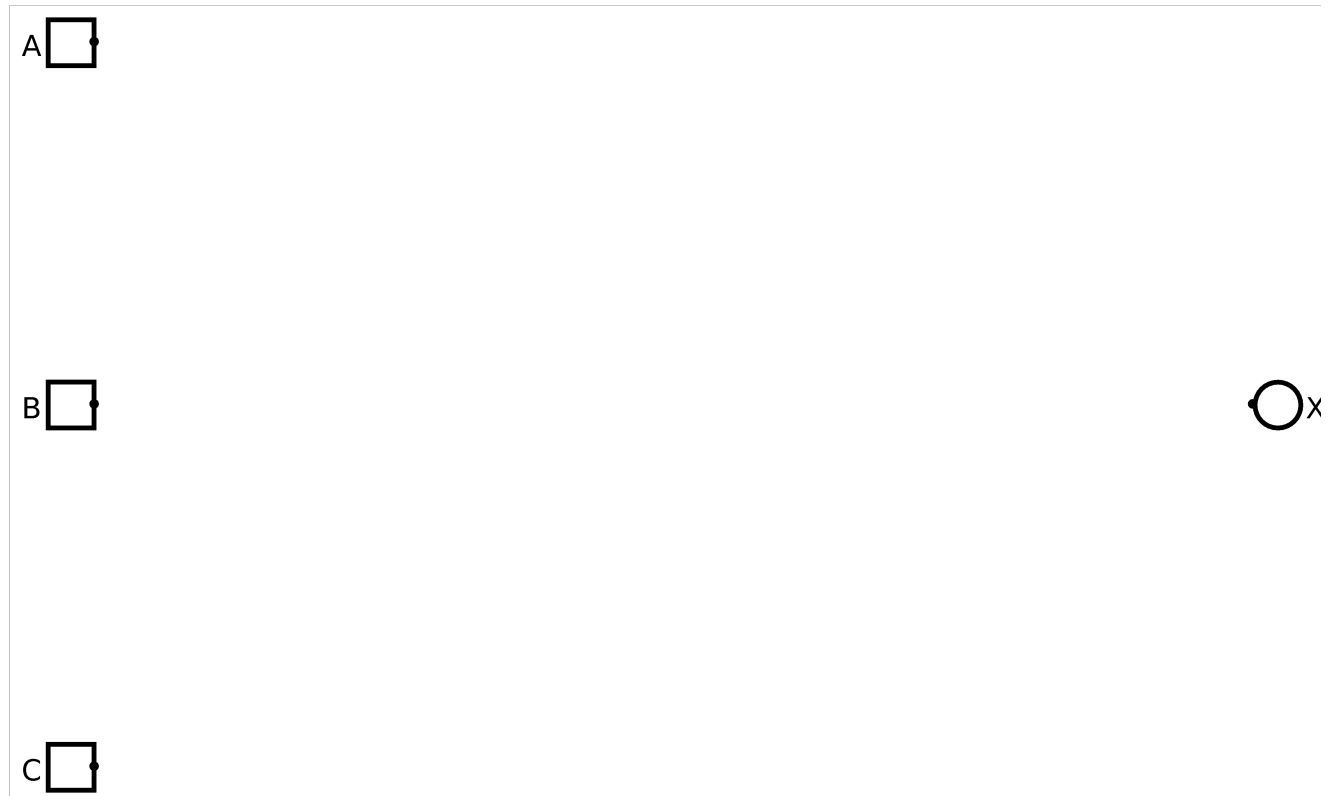
$$X = !(A \ \&\& \ B) \ \&\& \ !C$$

If you prefer the computer engineering style, you can think of the equation as

$$X = (A \ B)' \ C'$$

A	B	C	X
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

**Second**, draw a logic circuit (AND, OR, ...) to implement the boolean expression and corresponding truth table.



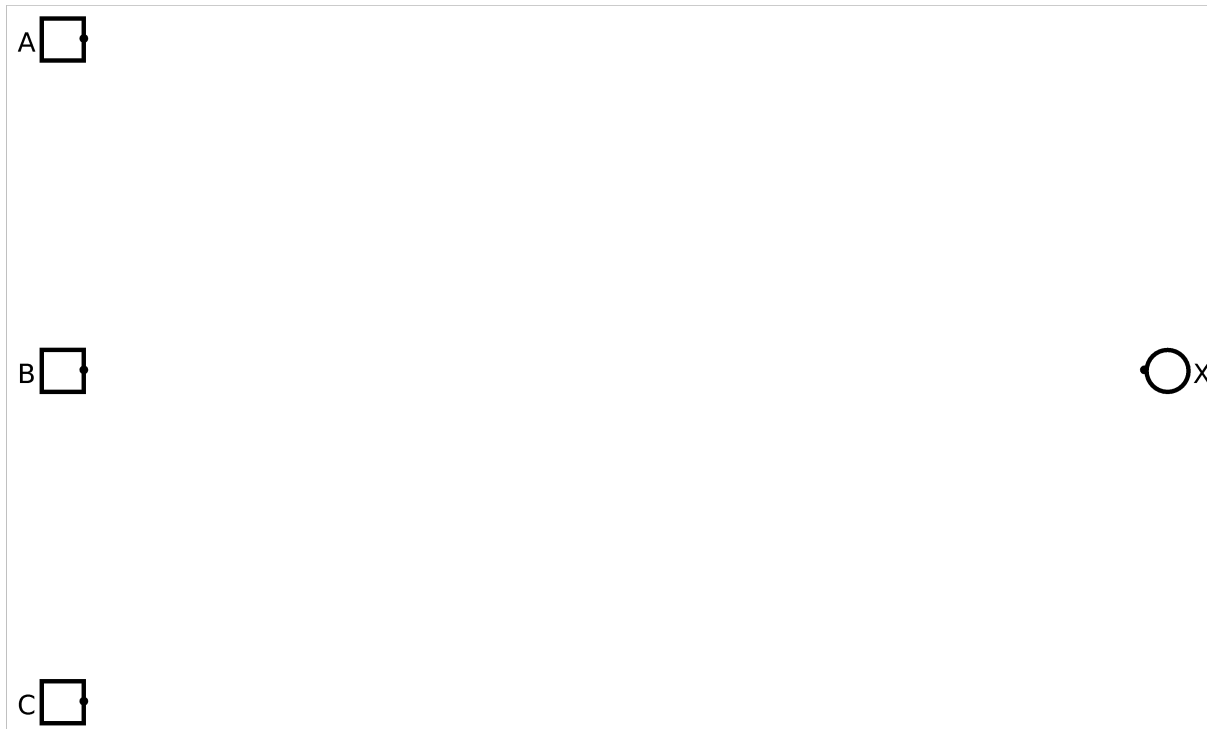
**Problem 11 (10 points) Truth table to Boolean expression and circuit**

The truth table below specifies a Boolean function with three inputs, **A**, **B**, and **C** and one output **X**.

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

**First**, write a Boolean expression corresponding to the function specified in the table. You do not need to write an “efficient” expression; however, ridiculously complex expressions will not be given full credit.

**Second**, draw a logic circuit (AND, OR, ...) to implement the boolean expression and corresponding truth table.



**Problem 12 (12 points)**

In this question, you are to fill in boxes representing the following C integer or pointer variables to show their values after each of seven sections of C code are executed. **You should consider all the sections as being independently executed after the following declaration and initialization statements:**

```
int    V[3] = {201, 235, 335} ;
int    *p = NULL ;
int    *q = NULL ;
```

As you might guess, `null` in Java is similar to `NULL` in C. Draw the value `NULL` with a little **X**. Don't ever just leave the pointer variable boxes empty.

<pre>p = &amp;V[1] ; q = &amp;V[2] ; *p = 150 ; *q = V[1] + 200 ;</pre>	<p>p <input type="text" value="X"/></p> <p>q <input type="text" value="X"/></p>	<table border="1"><tr><td>V[0]</td><td>201</td></tr><tr><td>V[1]</td><td>235</td></tr><tr><td>V[2]</td><td>335</td></tr></table>	V[0]	201	V[1]	235	V[2]	335
V[0]	201							
V[1]	235							
V[2]	335							
<pre>q = V ; p = q ; *p = 300 ; *q = 400 ;</pre>	<p>p <input type="text" value="X"/></p> <p>q <input type="text" value="X"/></p>	<table border="1"><tr><td>V[0]</td><td>201</td></tr><tr><td>V[1]</td><td>235</td></tr><tr><td>V[2]</td><td>335</td></tr></table>	V[0]	201	V[1]	235	V[2]	335
V[0]	201							
V[1]	235							
V[2]	335							
<pre>p = &amp;V[1] ; q = &amp;V[2] ; *q = *(p++) ; *p = 13 ;</pre>	<p>p <input type="text" value="X"/></p> <p>q <input type="text" value="X"/></p>	<table border="1"><tr><td>V[0]</td><td>201</td></tr><tr><td>V[1]</td><td>235</td></tr><tr><td>V[2]</td><td>335</td></tr></table>	V[0]	201	V[1]	235	V[2]	335
V[0]	201							
V[1]	235							
V[2]	335							
<pre>p = &amp;V[0] ; q = &amp;V[2] ; *q = (*p)++ ;</pre>	<p>p <input type="text" value="X"/></p> <p>q <input type="text" value="X"/></p>	<table border="1"><tr><td>V[0]</td><td>201</td></tr><tr><td>V[1]</td><td>235</td></tr><tr><td>V[2]</td><td>335</td></tr></table>	V[0]	201	V[1]	235	V[2]	335
V[0]	201							
V[1]	235							
V[2]	335							

# CSCI 255

## Handy Table of Numbers

### Powers of Two

$2^0$	1
$2^1$	2
$2^2$	4
$2^3$	8
$2^4$	16
$2^5$	32
$2^6$	64
$2^7$	128
$2^8$	256
$2^9$	512

$2^{10}$	1024
$2^{11}$	2048
$2^{12}$	4096
$2^{13}$	8192
$2^{14}$	16384
$2^{15}$	32768
$2^{16}$	65536
$2^{17}$	131072
$2^{18}$	262144
$2^{19}$	524288

$2^{10}$	1 Ki
$2^{20}$	1 Mi
$2^{30}$	1 Gi

### Hex table

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111