UNCA CSCI 235
Final Exam Spring 2018
May 8, 2018 3:00 pm – 5:30 pm

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 5:30 PM.

Name:_____

Problem 1: C expressions (10 points)

In the left column, there are twenty tricky and not-so tricky C expressions. Write their values in the right column. Express your answers in 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`.

`022`	
`0x22`	
`22 >> 3`	
`22 << 3`	
`22 / 3 * 5`	
`22 * 3 / 5`	
`22 & 7`	
`22 && 7`	
`22 | 7`	
`22 || 7`	
`22 ^ 7`	
`22 > 7`	
`~22`	
`!22`	
`22*0 && 10000/235`	

Problem 2: Range (2 points)
What is the range of values that can be stored in an 8-bit twos-complement numbers? (By the way, the `byte` of Java is an 8-bit twos-complement number.)

Problem 3: Decimal to two's complement conversion (4 points)
Convert the following four signed decimal numbers into **six**-bit *two's complement* representation. Some of these numbers may be outside the range of representation for **six**-bit two's complement numbers. Write "out-of-range" for those cases.

-10	**-1**
10	**40**

Problem 4: Q4.4 to decimal conversion (4 points)
Convert the following four Q4.4 *two's complement* numbers (four fixed and four fractional bits) into signed decimal representation.

00010001	**10001000**
11111111	**01000000**

Problem 5: Decimal to Q4.4 conversion (4 points)
Convert the following three 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.

-1.25
0.2
2.5

Problem 6: Floating point arithmetic (2 points)
I've tried both of these following C floating-point multiplications in gdb and one results in 1.0 and one does not:

```
5 * 0.2
2 * 0.5
```

Which is 1.0? Give an explanation for your choice.

Problem 7: Adding numbers with flags (8 points)
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).

010010 **+ 010010** **CF__, OF__, SF__, ZF__**	**110000** **+ 010000** **CF__, OF__, SF__, ZF__**
011111 **+ 110000** **CF__, OF__, SF__, ZF__**	**111000** **+ 110000** **CF__, OF__, SF__, ZF__**

Problem 8: CSCI arithmetic (4 points)
Perform the following operations and express the results as they should be for CSCI 235 and other geeky environments.

16 ki * 32
2 Mi / 8
$\log_2$(64 ki)

Problem 9: Expected time (2 points)
Assume that 80% of the time a requested memory access is found in the cache and retrieved in 20 n sec; and that the other 20% of the time, 40 µ sec is required. What is the average time required to perform a read request?

Problem 10: goto programming (8 points)
In the style of a recent homework, implement the C function shown below using only two control structures:

```
goto label ;
if (expression) goto label ;
```

This specifically means that you can't use the `for`, `while`, `switch`, `break`, `continue`, or even the statement block delimiters { and }. You can use the `if`, but only when the conditional expression is immediately followed by a `goto` statement. Also, do not use the `?:` operator of C (and Java) to simulate an `if-then-else`.

```
int huh(int p, int *d) {
    if (p > 0) {
      ++*d ;
    }
    return p + *d ;
}
```

Write your solution in the space below...

```
int huh(int p, int *d) {

}
```

Problem 11: Command line arguments (4 points)
This is very tricky! Suppose you have written the following program and compiled and linked it into an executable called `evilProg`.

```
#include <stdio.h>
int main(int argc, char *argv[]) {
  printf("%s %d %s%s\n",
         argv[1], argc-2, *(argv+2), argv[3]) ;
  return 0 ;
}
```

What command line arguments do you pass to `evilProg` to make it print this single output line?

won too 3 for

There is more than one correct answer.

`./evilProg` _____

Problem 12: C Programming (16 points)
Write a program that reads (**using `scanf`**) a sequence of time-of-day value from a terminated standard input stream. The times are entered in "traditional" US time-of-day format as shown below.

```
          12:10:00 AM     7:10:03 AM
                 12:05:10 PM    5:30:00 PM
```

Your output should be a neatly formatted list of ISO 8601 standard 24-hour clock times followed by a count of the number of clock times in the input stream. So, for the above example, the output should be:

```
      00:10:00
      07:10:03
      12:05:10
      17:30:00
Time read:    4
```

You may assume that the input contains properly formatted US time-of-day values but that white spaces (spaces, tabs, or new lines) can occur before and after the numbers and the `AM` and `PM` tokens. (This actually makes the programming task with `scanf` a little easier.)

Assume that 12:00:00 AM is midnight (00:00:00) and 12:00:00 PM is noon (12:00:00). That's what the US Government Printing Office decided in 2008. Also, it's the easiest to program.

Write your answer in the space below

```
// Your answer goes here.
#include <stdio.h>
int main(int argc, char *argv[]) {

}
```

Problem 13: Expression to truth table and circuit (8 points)

First, fill in the truth table on the right below so that it corresponds to the following Java (or C or C++) assignment:

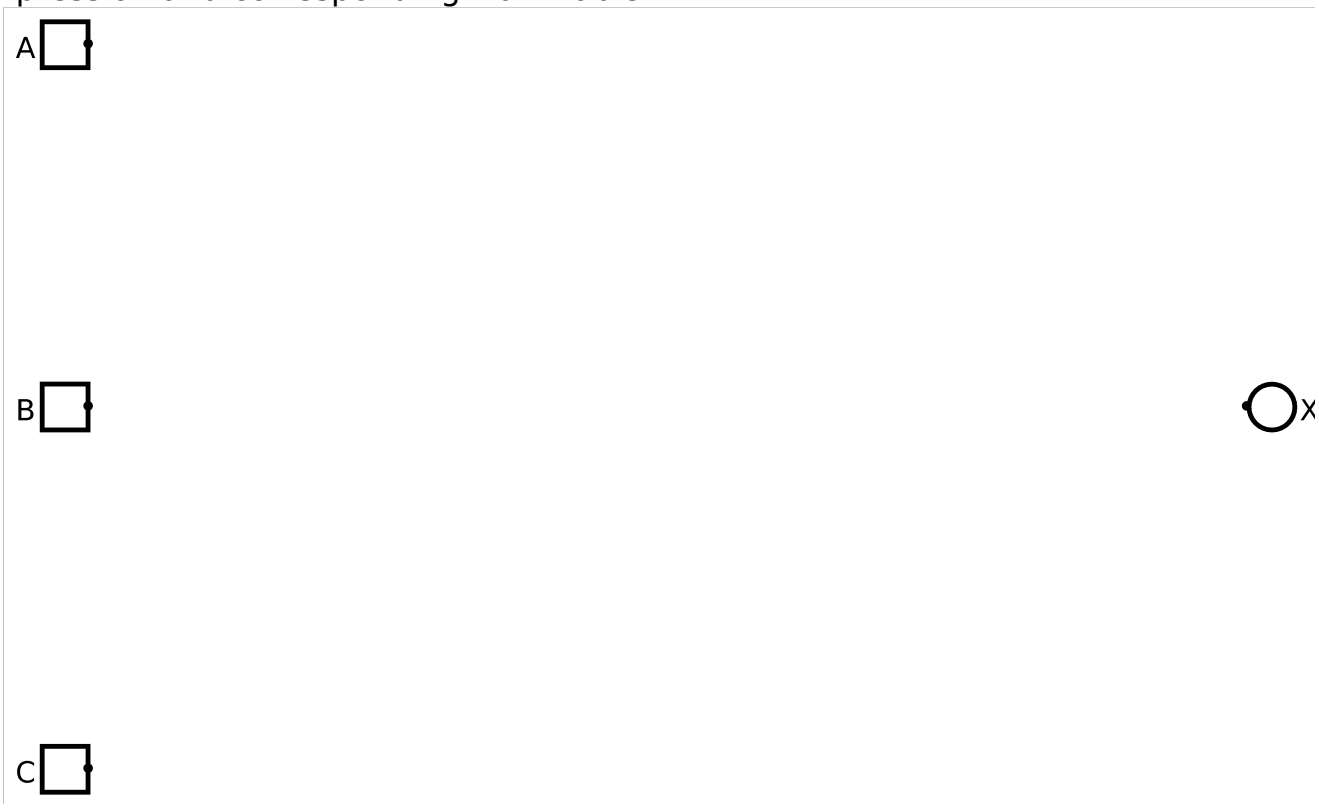
```
X = (A || !B) && C
```

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.



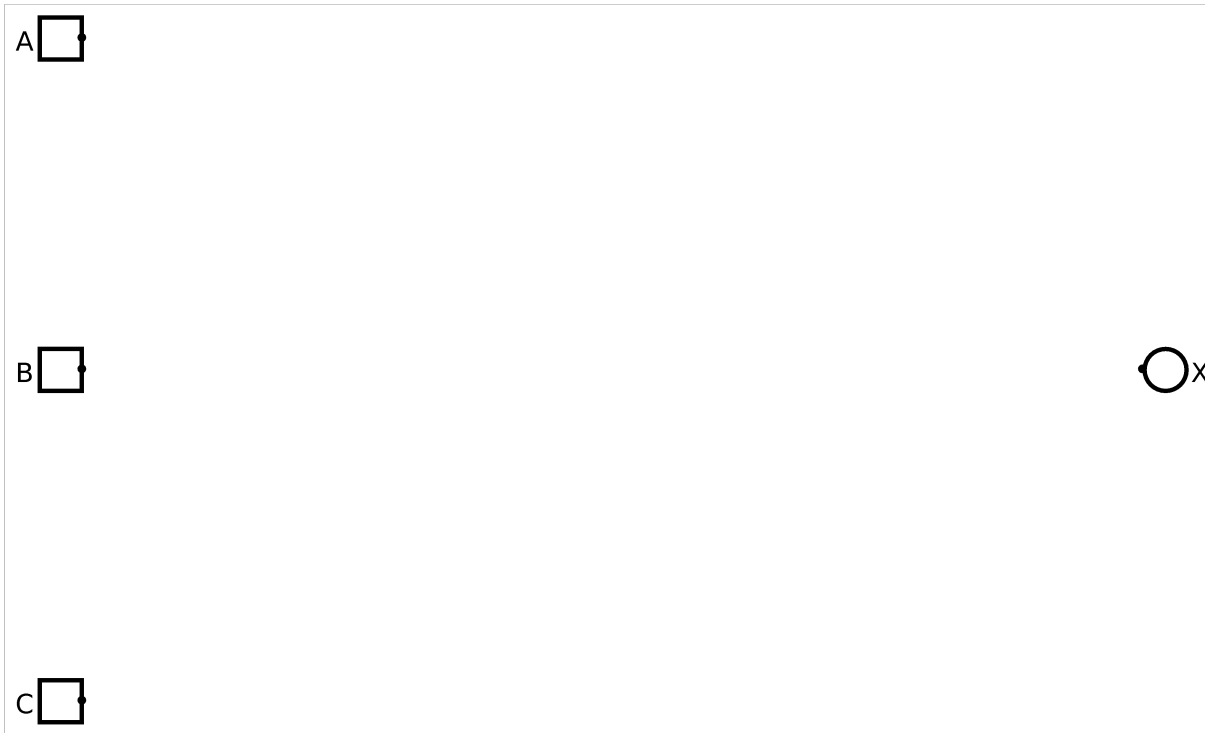
Problem 14: Truth table to expression and circuit (8 points)

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

P	Q	R	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
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.)

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

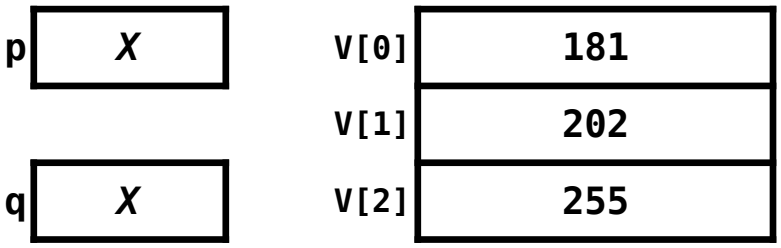


Problem 15: Pointers (8 points)
In this question, you are to fill in boxes representing the following C integer or pointer variables to show their values after each of seven sections of C code are executed. **You should consider all the sections as being independently executed after the following declaration and initialization statements**:

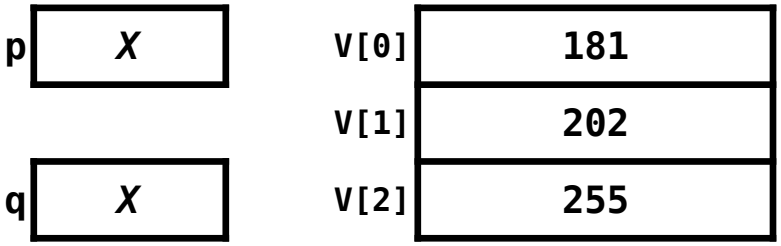
```
int    V[3] = {181, 202, 255} ;
int    *p = NULL ;
int    *q = NULL ;
```

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.

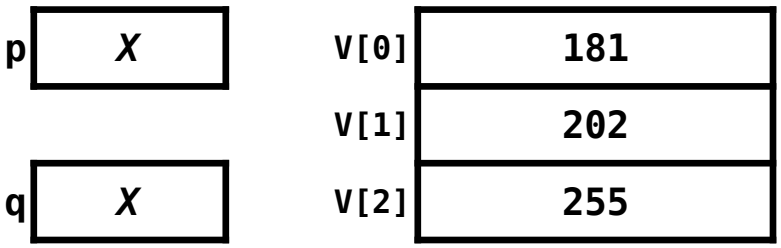
```
p = V ;
q = V+1 ;
*p = 200 ;
*q = 300 ;
```



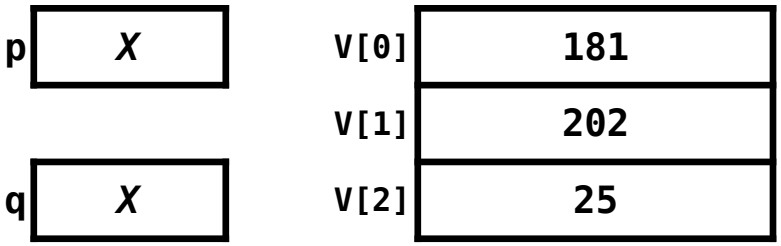

```
q = &V[1] ;
p = q++ ;
*p = *q ;
```




```
p = &V[0] ;
q = &V[2] ;
*p = *q – *p ;
*q = q - p ;
```




```
p = &V[0] ;
*(p++) = 235 ;
(*p) = 300 ;
```



Problem 16: Definitions (10 points)
To finish off, give short definitions of the following concepts, functions, hacks, types, variables, etc., as you have seen in this course (including its labs). *Feel free to skip two: I will grade the best six of eight definitions.*

`digitalWrite()`
x86-i64
Virtual memory
stack
`digitalRead()`
Randal E. Bryant
Raspberry Pi
current limiting resistor