UNCA CSCI 255
Final Exam Spring 2017
May 8, 2017 – 11:30 AM to 2:00 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.

If you want partial credit for imperfect answers, explain the reason for your answer!

Name:_____

Problem 1 (8 points) C expressions

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

`~5`	
`!5`	
`10 + 2`	
`10 & 2`	
`10 && 2`	
`10 | 2`	
`10 || 2`	
`10 ^ 2`	
`10 < 2`	
`10 << 2`	
`10 > 2`	
`10 >> 2`	
`10 / 2`	
`8 * 4 / 2`	
`8 / 4 * 2`	
`2015 * 17 || 13`	

Problem 2 (4 points) Decimal to two's complement conversion
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.

-32	32
10	-10

Problem 3 (4 points) Two's complement to decimal conversion
Convert the following four six-bit *two's complement* numbers into signed decimal representation.

101000	000000
111111	010111

Problem 4 (4 points) Range of numbers
Keep your answers simple for this question.

What is the range of numbers that can be stored in 8-bit two's complement numbers? (This is the same as the `byte` of Java.)

What is the range of numbers that can be stored in 8-bit unsigned numbers?

Problem 5 (4 points) Adding signed numbers
Add the following pairs of six-bit *two's complement* numbers **and indicate which additions result in an overflow by writing one of "overflow" or "no overflow" in each box**. You must write either "overflow" or "no overflow" in each box in addition to the result of the addition.

001111 + 011101	000101 + 010100
100011 + 100011	110101 + 110100

Problem 6 (6 points) Computer Science arithmetic
Perform the following operations and express the results as they should be for CSCI 255. Full credit given only for simple answers. Also, show your work.

64 ki * 64 ki
256 ki / 64 ki
$\log_2$(64 ki)

Problem 7 (5 points) Fixed point decoding
In the left column, are six-bit twos-complement ***fixed point*** numbers with three fractional bits (and consequently three integer bits). Express those numbers in standard decimal point notation in the right column.

100010	
100000	
000010	

Problem 8 (5 points) Fixed point encoding
In the left column are decimal numbers. Express these numbers as six-bit twos-complement fixed point numbers with three fractional bits (and consequently three integer bits) in the right column. *You may not be able to get an exact representation for all of them, but get as close as you can.*

3.25	
1.1	
-2.5	

Problem 9 (10 points) Implementing Boolean expressions
Complete the truth table on the right below so that it corresponds to the following Boolean equation

$$A = X Y + \overline{X} Z$$

If you prefer that your inversions be primes, you can think of the equation as

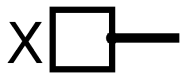
$$A = X Y + X' Z$$

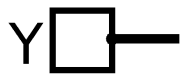
Or, if you really like Java and C expressions, you can go with

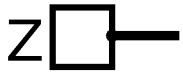
```
A = X && Y || !X && Z
```

X	Y	Z	A
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

On the remainder of this page, draw a logic circuit at the gate level that will implement the Boolean equation given above.







Problem 11 (10 points) Sigma notation to implementation
Complete the truth table for the Boolean function specified using Sigma notation:

F(X, Y, Z) = Σ(2, 3, 7)

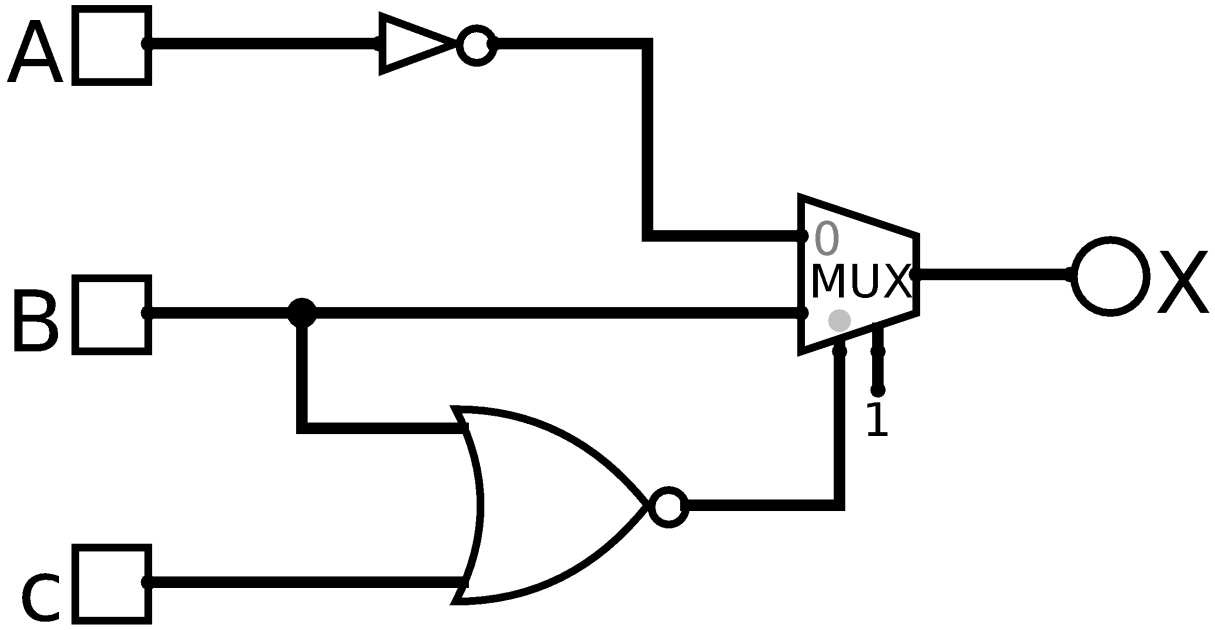
where A = F(X, Y, Z).

X	Y	Z	A
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Write a Boolean equation for the function shown above. **Don't try to simplify it. You'll only make the next step harder.**

And finally, implement your Boolean equation by adding and connecting **only** inverters **and** NAND gates in the following circuit.

Problem 12 (10 points) MUX Circuit to Boolean table and expression
Shown below is a digital circuit with inputs **A**, **B** and **C** and a single output **X**. The box labeled **MUX** is a multiplexer with two data inputs and one select input. (The 1 on is lower-right corner of the MUX is an enable input required by Logisim. You can ignore it.)



First, complete the truth table for the circuit.

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, write a boolean equation for the circuit.

Problem 12 (5 points) MIPS32 program
Implement the following C statement:

```
V[i] = 2017 + i ;
```

in MIPS32 assembly where `V` is stored in register `$a0` and `i` is stored in register `$a1`. Write the MIPS32 instructions required to implement this statement. (Hint: This is similar to an example done in the 29 March lab.)

Assume that `V` is an array of integers and that `i` is an integer.

Problem 13 (5 points) Memory cache
Suppose a computer has 64 Mi bytes of memory and 64 ki bytes of cache structured as an 8-way set associative cache with blocks (lines) of 64 bytes.

After figuring out the address size for the computer, draw a box for the typical address and divide it into tag, index, and offset fields. You have room to show your arithmetic. Use it!

Problem 14 (6 points) Disassembly required
For each of the nine 32-bit words shown below, translate the 32-bit word into the appropriate MIPS32 instruction. Use labels for instructions that are targets of branches. The last one has already been completed for you.

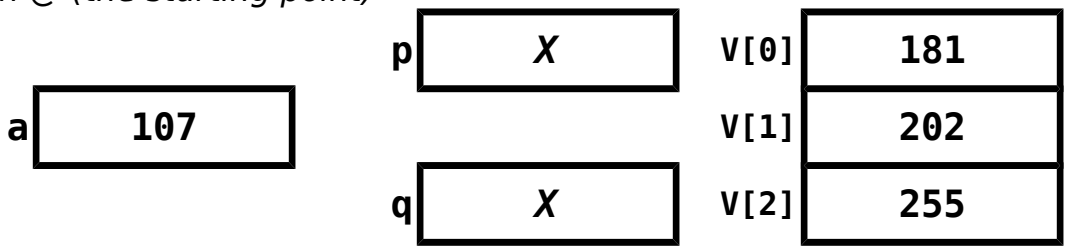
00010000000000000000000000000100
00100000000001000000000000000000
10001100100010000000000000000000
00000000100100000100000100000000
00100000101001011111111111111111
00011100101000011111111111111100
00100000100001000000000000000100
00000011100000000000000000001000
sll $zero,$zero,0 # nop 00000000000000000000000000000000

Problem 15 (4 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    a = 107 ;
int    V[3] = {181, 202, 255} ;
int    *p = NULL ;
int    *q = NULL ;
```

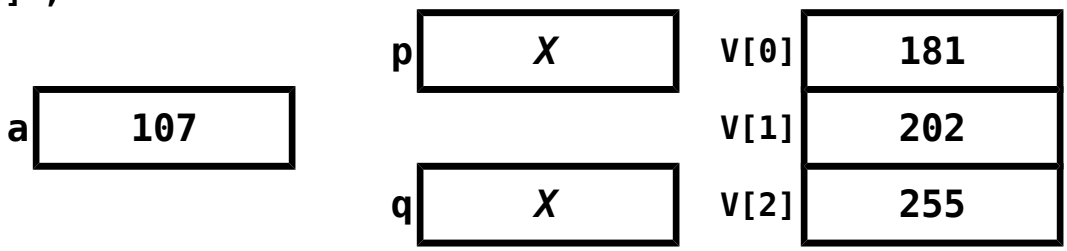
As you might guess, **`null`** in Java is similar to **`NULL`** in C. Draw the value **`NULL`** with a little ***X***. **Don't ever just leave the pointer variable boxes empty.**

Code section @ (the starting point)



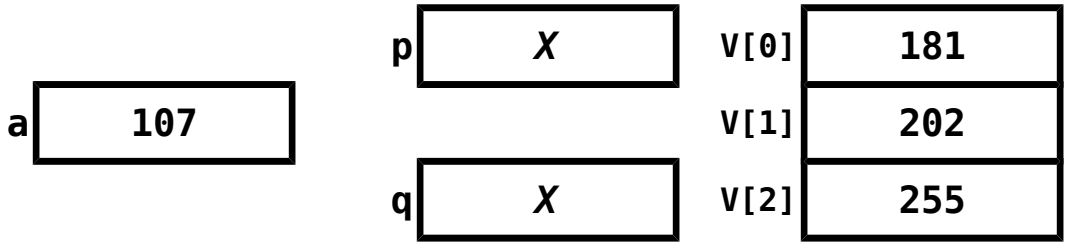
Code section A

```
p = &V[0] ;
q = &V[2] ;
*++p = *q ;
a = V[0] ;
```



Code section B

```
p = &V[1] ;
q = &V[0] ;
a = *q – *p ;
*q = q – p ;
```



Problem 16 (6 points) goto-full programming
In this problem, you are going to implement the following C function:

```
unsigned int collatz(unsigned int n) {
  unsigned int count = 1 ;
  while (n != 1) {
    if (n && 1 == 1) {
      n = 3 * n + 1 ;
    } else {
      n = n / 2 ;
    }
  }
  return count ;
}
```

If the style of Exam 2, implement the `collatz` function as a C function that only uses two control structures:

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

Do not use the `?:` operator of C (and Java) to simulate an `if`-`then`-`else`. 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.

```
unsigned int collatz(unsigned int n) {
  unsigned int count = 1 ;
```

```
  return count ;
```

Problem 17 (4 points)
Describe how the following have been used in the CSCI 255 labs. Your four best descriptions will be graded. The other four will be ignored. Keep it short and precise

a) **`adduser`**

b) current limiting resistor

c) **`DigitalWrite()`**

d) I^2C (inter-integrated circuit) bus

e) LSM303 accelerometer

f) Pi Cobbler

g) **`PORTB`** – a special function register

h) wave – SystemVerilog