UNCA CSCI 235
Final Exam Spring 2019 answers
6 May 2019 – 3:00 pm to 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.

Name:______________________________

Problem 1 (10 points) C expressions
In the left column, there are fifteen 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`.

`0353`	235
`0xC8`	200
`11 && 0`	0
`11 || 0`	1
`20 & 11`	0
`20 | 11`	31
`20 ^ 11`	31
`20 / 11`	1
`20 + ~11`	8
`22 << 2`	88
`22 >> 2`	5
`3 * 4 / 5`	2
`(3 * 4) / 5`	2
`3 * (4 / 5)`	0
`(23 * 33) && (0 * 14)`	0

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.

1 **000001**	**32** **out-of-range**
-1 **111111**	**-32** **100000**

Problem 3 (3 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.

10101010 **-5.375**	**01010101** **5.3125**

Problem 4 (3 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.

2.35 **00100110**	-1.25 **11101100**

Problem 5 (6 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).

111010 + 000110 000000 CF 1, OF 0, SF 0, ZF 1	011010 + 000110 100000 CF 0, OF 1, SF 1, ZF 0
101010 + 101010 010100 CF 1, OF 1, SF 0, ZF 0	110110 + 110110 101100 CF 1, OF 0, SF 1, ZF 0

Problem 6 (2 points) Range 1
What is the range of numbers that can be stored in 16-bit twos-complement numbers? (The `int` of Arduino C++ is a 16-bit twos-complement number.)

-32768 to 32767

Problem 7 (2 points) Range 2
What is the range of numbers that can be stored in 8-bit unsigned numbers? (The `unsigned char` of Arduino C++ is an 8-bit unsigned number.)

0 to 255

Problem 8 (6 points) Formatted printing

Suppose that the `int` variable C has the value 140 (in decimal). The left column in the table below has a `printf` statement. The right column has the desired output for that `printf` within a six character field. Your task is to fill in the underlined part (the stuff after the %). ***You must use a single "conversation specifier" (the thing starting with a %) in your format string. No "ordinary characters" are allowed.*** This means the following are not allowed because they contain ordinary characters.

```
printf("000140", C) ;   // contains only ordinary characters
printf("   %3d", C) ;   // starts with three ordinary characters
```

`printf("%6X",C) ;`	_ _ _ _ 8 C
`printf("%6d",C) ;`	_ _ _ 1 4 0
`printf("%6x",C) ;`	_ _ _ _ 8 c
`printf("%6o",C) ;`	_ _ _ 2 1 4
`printf("%+6d",C) ;`	_ _ + 1 4 0
`printf("%06d",C) ;`	0 0 0 1 4 0

Problem 9: goto programming (8 points)

In the style of a recent lab, 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 big_letter_count(const char *s) {
    int n = 0 ;
    while (*s != 0) {
        if ('A' <= *s && *s <= 'Z') {
            ++n ;
        }
        ++s ;
    }
    return n ;
}
```

```
int big_letter_count(char *s) {
    int n = 0 ;
      goto loopTest ;
loopStart:
      if (!('A' <= *s && *s <= 'Z')) goto noIncN ;
      ++n ;
noIncN:
      ++s ;
loopTest:
      if (*s != 0) goto loopStart ;
   return n ;
}
```

Problem 10 (6 points) Strings in C

A Java or Python programmer might be puzzled by the absence of a `length()` method or a `len()` function for determining the length of a character string.

Rewrite the big_letter_count program to use a C **for** loop while using **s** as a character array indexed by a variable **i**. That is, fill in the blanks to make your program look more like a Java program. However, you still can't use **length**! That is not in C.

```
int big_letter_count(const char s[]) {
    int n = 0 ;

    for(int i = 0 ; *s != '\0' ; ++i ) {

        if ('A' <= s[i] && s[i] <= 'Z') {
           ++n ;
        }
    }
    return n ;
```

Problem 11 (6 points) CSCI arithmetic

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

32 * 128 Gi = $2^5 * 2^7 * 2^{30}$ = 2^{42} = 4 Ti
4 Mi / 8 = $2^2 * 2^{20} / 2^3$ = 2^{19} = 512 ki
$\log_2$(8 Gi)=$\log_2$($2^3 * 2^{30}$)=$\log_2$(2^{33})= 33

Problem 12 (13 points): C Programming

Write a program that reads from standard input a sequence of pairs of county names (15 characters or less) and their populations and prints a nicely formatted list of the input pairs, in the order they were read, along the average population of the counties. So, if the input to your program is something like:

```
Buncombe 257607    Haywood
  61084  Transylvania   33956
```

Your program output should resemble the following:

```
Buncombe       257607
Haywood         61084
Transylvania    33956
AVERAGE:       117549
```

```c
#include <stdio.h>
int main(int argc, char *argv[]) {
  int  totalPeoples = 0 ;
  int  countCounties = 0 ;
  char county[16] ;
  int  peoples ;
  while (scanf("%15s %d", county, &peoples) == 2) {
    totalPeoples = totalPeoples + peoples ;
    ++countCounties ;
    printf("%-20s %8d", county, peoples) ;
  }
  int average = totalPeoples / countCounties ;
  printf("AVERAGE:           %8d", average) ;
}
```

Problem 13 (5 points) Boolean expression to truth table
Fill in the truth table on the right below so that it corresponds to the following Java (and C) expression:

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

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

```
X = A' (B + C) + A B C
```

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	0
1	1	0	0
1	1	1	1

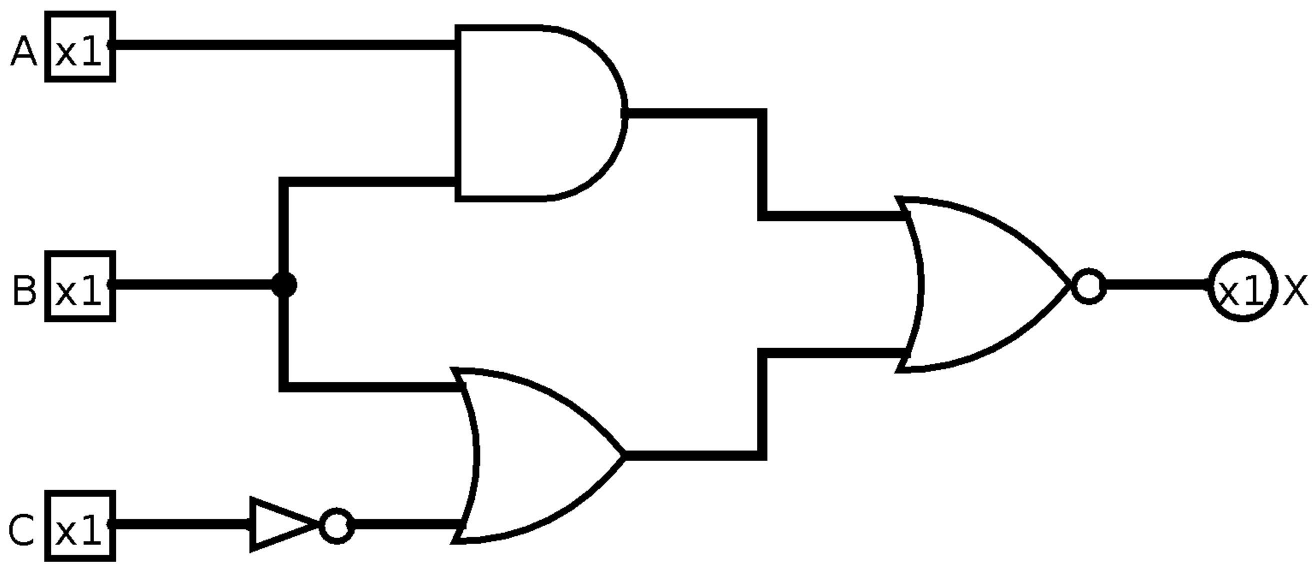
Problem 14 (5 points) Truth table to Boolean expression
The truth table below specifies a Boolean function with three inputs, **A**, **B**, and **C** and one output **X**.

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

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. The phrase "ridiculously complex expressions" means "expressions with require more than five minutes of instructor time to decode".

A' B' C + A' B C' + A B' C' + A B C'
A' B' C + A B' C' + B C' ***simplified***

Problem 15 (8 points) Circuit to Boolean expression and truth table
A gate-level circuit is shown below with three inputs on the left and a single output on the right.



First, write the Boolean expression corresponding to this circuit. (Don't worry about the "x1". It indicates that the connection is for a single bit.)

(A B + B + C')'

ECE 209/MATH 251: (A B + B + C')' → (B + C')' → B' C

Next, complete the following truth table so that it corresponds to this digital logic circuit.

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

Problem 16: Definitions (7 points)
Give short definitions of the following concepts, functions, hacks, programs, types, variables, etc., you have seen in the labs and homework of this course, *Feel free to skip one: I will grade the best seven of eight definitions.*

`330 Ω`
`bit banging`
breadboard
CircuitPython
current limiting resistor
nano
`os.walk()` and/or `nftw()`
Tinkercad circuits

Problem 17 (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] = {201, 235, 335} ;
int    *p = NULL ;
int    *q = NULL ;
```

As you know, **`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 ;
```

p	&V[0]	V[0]	200
		V[1]	300
q	&V[1]	V[2]	335

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

p	&V[1]	V[0]	201
		V[1]	335
q	&V[2]	V[2]	335

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

p	&V[0]	V[0]	2
		V[1]	235
q	&V[2]	V[2]	335

```
p = &V[0] ;
q = &V[1] ;
*(++q) = ++(*p);
```

p	&V[0]	V[0]	202
		V[1]	235
q	&V[2]	V[2]	202

CSCI 235
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