UNCA CSCI 255
Final Exam Fall 2010
9 December, 2010

This is a closed book and closed notes exam. It is to be turned in by 3:30 pm. Calculators, PDA's, cell phones, and any other electronic or communication devices may not be used during this exam.

Name:_____

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

Problem 1 (1 point)
Complete the following C statement so that the variable `myName` points to a character string containing your *clearly spelled* name.

```
char *myName = "                                    " ;
```

Problem 2 (3 points)
In the nine boxes below are nine possible names for C identifiers. Cross out the ones that would **not** be legal C identifier names. (The answers would be the same for Java identifiers.)

`double8`	`december_8`	`$dec8`
`w`	`Dec8`	`_dec8`
`double`	`"December 8"`	`8Dec2010`

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

18	**-1**
-24	**31**

Problem 4 (6 points) Two's complement to decimal conversion
Convert the following six six-bit two's complement numbers into signed decimal representation

111100	**000111**
010101	**111000**
100001	**100111**

Problem 5 (6 points) Adding
Add the following pairs of six-bit two's complement numbers **and indicate which additions result in an overflow**.

011101 **+ 101010**	**001010** **+ 011100**
111111 **+ 111111**	**001101** **+ 001011**
101101 **+ 101010**	**001010** **+ 111100**

Problem 6 (3 points) Ranges
What is the number of different values that can be represented by 7 binary digits?

What is the greatest number that be represented in 7-bit two's complement notation?

What is the smallest number that can be represented in 7-bit two's complement notation?

Problem 7 (4 points) Memories
A 8 MB memory has a 32-bit word size. How many words are contained in this memory?

A memory has 2k words. Each word contains 16 bits. How many bytes are contained in this memory?

How many bits are required to address the a memory of 8G words?

Problem 8 (3 points) Bitwise operations
Perform the following bit-wise logical operations on 8-bit numbers expressed as two hexadecimal digits. Your answer should also be expressed in hexadecimal.

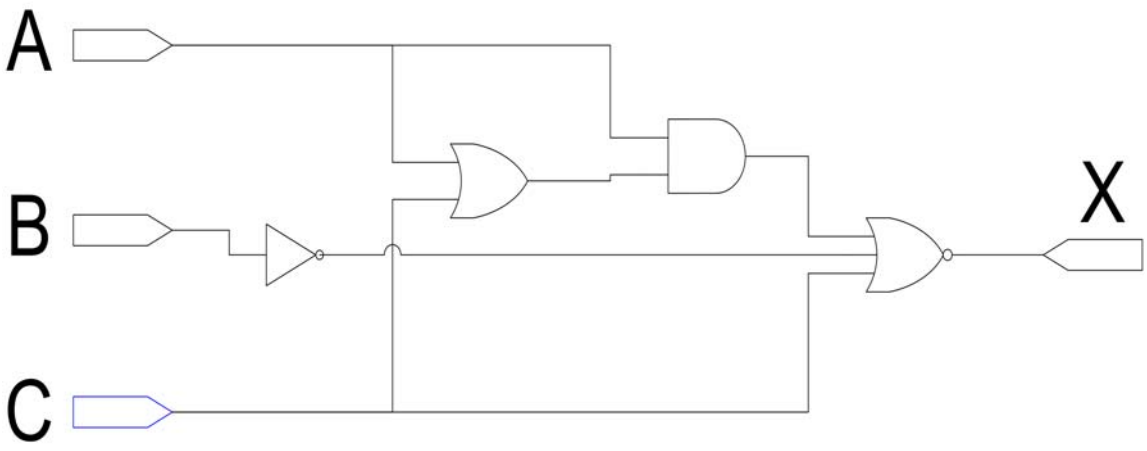
AND(25,A9) -->
OR (25,A9) -->
NOT (A9) -->

This is the same as the Java operations `0x25 & 0xA9`, `0x25 | 0xA9`, and `~0xA9`.

Problem 9 (10 points) Truth to Gates
Draw a circuit, at the gate level, that will implement the following truth table, where A, B, C, and D are inputs and where Z is the single output.

A	B	C	D	Z
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Problem 10 (6 points) Gates to truth
Give the truth table for the gate-level circuit shown below. The three inputs are on the left, the output is on the right.



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	

Problem 10 (4 points)
Describe how the following `while` loop can be expressed in C without using the `while` or the `for`. Instead use the `if` and the `goto` and a couple of labels.

```
while (n > 0) {
     n = n - 5 ;
}
```

Problem 11 (4 points)
Describe how the following nested `while` loop can be expressed in C without using the `while` or `for`. Instead use the `if` and the `goto` and a couple of labels.

```
while (k < 75) {
     j = k + 15 ;
     while (j > 33) {
          j = j – 4 ;
     }
     k = 100 – j ;
}
```

Problem 12 (4 points)
Describe how the following `if` statement can be expressed in C without using the `&&` or `||` operators. Be sure that the second operand of the `&&` or `||` is evaluated only when the C evaluation rules say it should be.

```
if (n != 0 && (x != 1 || 3/(x-1) == 7)) {
     n = 5/n ;
}
```

Problem 13 (4 points)
Describe how the following `if-else` statement can be expressed in C without using the keyword `else`. Instead use the `if` and the `goto` and a couple of labels.

```
if (n < 0) {
     n = n + 5 ;
} else {
     n = n – 5 ;
}
```

Problem 14 (4 points)
Suppose `x`, `y`, and `z` refer to a file registers (data memory locations) on a PIC24, write the PIC24 assembly code needed to execute the following C statement.

```
x = x – (y+z)*3 + 200 ;
```

Problem 15 (4 points)
Suppose `x` refers to a file register (data memory location) on a PIC24, write the PIC24 assembly code which sets register `W5` to the absolute value of `x`, that is, which executes the following C sequence.

```
if  (x < 0{
     W5 = -x ;
} else {
     W5 = x ;
}
```

Problem 16 (4 points)
Suppose `x` refers to a file register (data memory location) on a PIC24, write the PIC24 assembly code which executes the following C sequence.

```
while (x < 105) {
     x = x + 10 ;
}
```

Problem 17 (8 points)
Assume that PIC registers W0 to W3 and W15 contain the following values:

Register	Value
W0	x9002
W1	x9008
W2	x0007
W3	xBEEF
W15	x9006

And assume memory locations (file registers) x9000 to x9008 contain the following values.

Memory	Value
x9000	x0011
x9002	x0005
x9004	x9000
x9006	x0003
x9008	x9008

For each of the following eight PIC instructions describe (1), the registers or memory locations changed by those instructions and (2), the new values of the changed registers or memory locations. (You may omit any changes made to the SR file register.)

Assume that the instructions are executed independently. That is, they are not executed in order.

`mov W0, W3`
`mov [W0], W3`
`mov [++W0], W3`
`mov [W0++], W3`
`mov [W1], [W0]`
`mov [W1++],W1`
`push W3`
`pop W3`

Problem 18 (8 points)
In the boxes below there are eight PIC24 instructions on the top line and 24 underscores on the bottom. Fill in those 24 underscores with the 24 binary bits needed to encode each instruction in PIC24 instruction set.

`add SR` _
`add #10, W7` _
`add W5, #10, W7` _
`mov WREG, SR` _
`mov #30, PCH` _
`mov [W5],W6` _
`mov WREG, TRISB` _
`mov W0, SR` _

Problem 19 (8 points)
And finally here is a little section of C code to get you started.

```
int  j ;
int  k ;
int *p ;
int *q ;
int  v[2] ;

j    = 15 ;
k    = 25 ;
v[0] = 35 ;
v[1] = 45 ;
// The following line prints
//   15, 25, [35, 45]
printf("%d, %d, [%d, %d]\n", j, k, v[0], v[1]) ;
```

Now explain what values are printed by the remaining C. A little drawing will help.

```
p = &j ;
q = &k ;
*p = *q ;
++*q ;
printf("%d, %d, [%d, %d]\n", j, k, v[0], v[1]) ;
```

```
p = q ;
q = &V[0] ;
*p = *q ;
++*q ;
printf("%d, %d, [%d, %d]\n", j, k, v[0], v[1]) ;
```

```
p = &V[0] ;
q = &V[1] ;
++p ;
++*p ;
++*q ;
printf("%d, %d, [%d, %d]\n", j, k, v[0], v[1]) ;
```