Final Exam CSCI 255 Spring 2002
Open Book Section
13 May 2002

Name: _____

This exam has a closed book and an open book section. Until you have turned in the closed book section, you are not allowed to use any resource materials: books, notes, or calculators.

Be sure to show your work in order to get full credit for the problem. When possible place your answers in the provided space. There are 5 questions for a total of 56 points on the three-page open book section of the quiz.

This exam is to be turned in by 8:30 pm.

Problem 1 (8 points)

Compute the following bit-wise logical operations on 4-bit binary numbers.

0101 AND (0111 OR 1111)	(NOT 0011) OR (NOT 0101)

Problem 2 (8 points)

Implement the truth table shown below as circuit. The "inputs" to the truth table are A, B, and C. The output is Z. You may implement the circuit using any of the various digital logic structures given in chapter 3 of the textbook.

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

Problem 3 (20 points)

Write five separate LC/2 assembly programs to compute the following five C statements:

`R2 = 2*R3 + 20 ;`	
`if (R3 > R2)` `     R2 = R3 ;`	
`R3 = R2 & R3 ;`	
`while (R2 > 0)` `     R2 = R2 - 5 ;`	

`R5 = H[R2] + H[R3] ;`	`          ST   R0,SaveR0`
	`          LEA  R0,H`
	
	`          LD   R0,SaveR0`
	`SaveR0    .BLKW     1`
	`H         .BLKW     100`

Problem 4 (12 points)

Assuming that `B255` is a function that receives a single integer as an argument and then returns a single integer and, consequently, has the following prototype:

```
int B255(int n) ;
```

implement the following rather silly C function in LC/2 assembler:

```
int A255(int n) {
     return B255(n) + 3 ;
}
```

Problem 5 (8 points)

Once upon a time, some graduating seniors were required to translate the following C function into LC-2 assembler:

```
int Fin255(int *Px, int y) {
     int r ;
     r = *Px ;
     *Px = r + y ;
     return r ;
}
```

One student proposed the following solution:

```
STR     R7,R6,#1        ; Store return address

LDR     R0,R6,#3        ; RO <- Px
LDR     R2,R0,#0        ; R2 = *Px
STR     R2,R1,#0        ; r(R1) = *Px
LDR     R3,R6,#4        ; R3 <- y
ADD     R4,R1,R3        ; R4 = R1(R which equals *Px) + R3(y)
STR     R4,R2,#0        ; R2(*Px) = R4(r+y)
STR     R1,R6,#0        ; return r

LDR     R7,R6,#1        ; load return address
LDR     R6,R6,#2        ; restore dynamic link
```

There's at least one bug in this program. Fix the program by correcting it bugs.