CS 61C (Clancy)

Exam 2

Problem 1 (4 points, 8 minutes)

Given below is a MAL program segment that computes (x+1.0)^2 by adding x^2 to 2x, then adding 1 to that sum.

```
        .data
x:      .float
answer: .float
one:    .float  1.0
        .text
_start:
        l.s     $f4,x
        l.s.    $f6, one
        mul.s   $f8, $f4, $f4     # x^2
        add.s   $f8, $f8, $f4     # + 2*x
        add.s   $f8, $f8, $f4
        add.s   $f8, $f8, $f6     # + 1.0
        s.s     $f8, answer
```

Part a

Consider the case where x is 2.0^12. What is the difference between the value stored in answer and the actual value of (2.0^12 +1.0)^2 ? (If the answer is the computed correctly, the difference will be 0.) Show your work.

Part b

Does the sequence in which the terms are added affect the correctness of the answer? Briefly explain.

Problem 2 (6 points, 10 minutes)

Translate the C/C++ function PrintDownUp to MAL, retaining its recursive structure, passing its argument in the appropriate register, and following the usual register conventions. Translate putchar into a putc pseudoinstruction whose register argument contains the character to print.

```
void PrintDownUp (char c) {
   if (c=='a') {
      putcher (c);
```

```
    } else {
        putchar (c);
        PrintDownUp (c-1);
        putchar (c);
    }
}
```

Problem 3 (8 points, 15 minutes)

An addendum to this exam contains the code for iout.soln.s, the solution to the first part of project 3, modified slightly as follows. The infinite loop

```
loop:
        la      $4, string
        jal     print
        j       loop
```

has been replaced by five calls to print, followed by another loop that does nothing 10000 times, followed by done. When spim is started* and the program modified as just described is loaded and run, the output

```
Just  wasting   time
Just  wasting   time
Just  wasting   time
Just  wasting   time
Just  wasting   time
```

(95 characters - there's a carriage return and a line feed at the end of each line) is produced and the program terminates.

Two of the instructions in the intrp routine are boxed:

- the instruction sw $0,8 ($10) immediately preceding the branch to intDone;
- the instruction addiu $8,$8,1 six lines further on.

In this problem., you consider the effect of removing each boxed instruction.

*Making spim work correctly requires that it be supplied with command-line options-memio and -quiet after giving the command stty min 1 -icanon to the UNIX shell. We assume here that spim has been started in this way.

Part a

What will be the effect of removing the boxed sw $0,8($10) and running the program? Describe the program's output. Also indicate whether the program terminates normally or ends up in an infinite loop and, if so, where. Briefly explain your answer.

Part b

What will be the effect of removing the boxed addiu $8,$8,1 (retaining the sw) and running the program? Describe the program's output. Also indicate whether the program terminates normally or ends up in a infinite loop and, if so, where. Briefly explain your answer.

Problem 4 (6 points, 10 minutes)

Consider a logic circuit that, given inputs X0, X1, and X2, produces a binary encoding in outputs q1 and q0 of how many of the Xk are 1. A truth table relating q1 and q0 to the Xk appears below.

X0	X1	X2	q1	q0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Using and, or, not, and xor, design Boolean equations to represent the circuit. your equations should be simplified where possible; show your work.

Problem 5 (4 points, 6 minutes)

Consider a function IsolateFloatFields that isolates components of a normalized positive floating point value in IEEE 32-bit format. Given such a value, IsolateFloatFields should return

a. the exponent, and
b. the integer that results from omitting the binary point from fraction represented by the significand.

For example, if the value 2.875 base 10 (which 1.0111 * 2^1) is passed to IsolateFloatFields, it should return the integer 1 for the exponent and the integer whose binary representation is 10111 followed by nineteen zeroes for the significand.

Complete the assignment statements in either the C version or C++ version (not both) of the function IsolateFloatFields below. Don't add any code outside the blanks.

The TheBits functions returns an unsigned integer whose bits are the same as those of its float argument. It's needed since bitwise operators in C and C++ may not be applied to float values.

C version

```
void IsolateFloatFields (float x, int *exponent, int *fractBits) {
    unsigned int bits = TheBits (x);

    *exponent = ________________________________________;
    *fractBits = ________________________________________;
}
```

C++ version

```
void IsolateFloatFields (float x, int &exponent, int &fractBits) {
    unsigned int bits = TheBits (x);

    exponent = ________________________________________;
    fractBits = ________________________________________;
```