CSE 31

Midterm 1 Sample

Time : 75 minutes

Name:

1: Number Representation

a) Fill in the following table :

Decimal (Base 10)	Binary (Base 2)	Hexadecimal (Base 16)
	1	
		0x3
101		
	0001 0010	
		0xC953
131071		

b) Fill in the following table :

Binary	Unsigned	Signed	1's Complement	2's Complement	Biased
0000 1111					
0101 0101					
1010 1010					
1111 1111					

c) Fill T/F in the following table :

Property	Unsigned	Signed	1's Comp	2's Comp	Biased
Can represent positive numbers					
Can represent negative numbers					
Has more than one representation for 0					
Use the same addition process as unsigned					

d) What is the value in decimal of the most negative 16-bit 2's complement integer?

e) What is the value in decimal of the most positive 16-bit signed integer?

2: C Code

The following program is compiled and run on a MIPS computer.

```
1       int main() {
2               int i;
3               int four_ints[4];
4               char* c;
5
6               for(i=0; i<4; i++) four_ints[i] = 2;
7
8               c = (char*)four_ints;
9               for(i=0; i<4; i++) c[i] = 1;
10
11              printf("%x\n", four_ints[2]);
12      }
```

a) What does it print out? (The “%x” in printf is used output a word in hexadecimal format.)

b) If we change the 2 on line 11 to a 0, then recompile and run, what would be printed? (hint: Consider how many hex digits are in an int and in a character, i.e. not the same as bytes)

c) The following function should allocate space for a new string, copy the string from the passed argument into the new string, and convert every lower-case character in the **new** string into an upper-case character (do not modify the original string). Fill-in the blanks and the body of the for() loop:

```
char* upcase(char* str) {
        char* p;
        char* result;

        result = (char*) malloc(_____________________________);

        strcpy(_____________,_____________);

        for( p=result; *p!='\0'; p++ ) {
        /* Fill-in 'A' = 65, 'a' = 97, 'Z' = 90 , 'z' = 122 */




        }
        return result;
}
```

d) Consider the code below. The upcase name() function should convert the i^{th} name to upper case by calling upcase by ref, which should in turn call upcase(). Complete the implementation of upcase_by_ref. You may not change any part of upcase_name.

```
        void upcase_by_ref( char** n ) { /* Fill-in */



        }

        void upcase_name(char* names[], int i) { /* No not touch */
                upcase_by_ref( &(names[i]) );
        }
```