EECS 281: Homework #3 Due: Tuesday, 28.09.2004

0. Practice the wakerly problems (see website soln.) and do not hand in: 2.1(a,b,e), 2.7a, 2.9a, 2.10a, 2.12a, 2.37.

1. Using standard C++ precision and data types, convert the following into two’s complement big-endian binary and if not, then show why not?:

unsigned char x = ’A’;	01000001	unsigned char x = 0255;	
unsigned char x = 0x255;		unsigned char x = 255;	
signed char x = 255;		unsigned char x = 0128;	
unsigned char x = 128;		unsigned char x = 0xfa;	
unsigned char x = 35;		unsigned char x = -35;	
signed char x = 127;		signed char x = 128;	
signed char x = -128;		signed char x = -0x2;	
signed char x = -07;		signed short x = -2;	
signed short x = 35;		signed short x = -35;	
signed short x = ’a’;		signed short x = -’a’;	

2. Assume VHDL data types convert the following into two’s complement big-endian binary:

signal x: std_logic_vector(4 downto 0):= b”10111”;	
signal x: std_logic_vector(0 to 4):= b”10111”;	
signal x: std_logic_vector(7 downto 0):= o”45”;	
signal x: std_logic_vector(0 to 7):= x”ab”;	

3. Using C++/C#/Java operator precedence, add the correct parenthesis (signed int a, b, ..., w, x, y, z;):

<table>
<tr><td>a = w & x & y & z;</td><td>a = w | x | y & z;</td></tr>
<tr><td>a = ~ x & y & ~ z;</td><td>a = x | y & z;</td></tr>
<tr><td>a = x | y ^ w & z;</td><td>a &= x & y & z;</td></tr>
<tr><td>a = x * y + z;</td><td>a = z + y * z;</td></tr>
<tr><td>a = z + y * z % w / v - c;</td><td>a = x & y | z;</td></tr>
<tr><td>a = z | y & z;</td><td>a = ~ z ^ y & z;</td></tr>
<tr><td colspan="2">a += b + c >> d & e ^ f | ~ g % h * i - j;</td></tr>
</table>

4. Using VHDL operator precedence, add the correct parenthesis:

   a <= b + c SRL d AND e XOR f OR NOT g MOD h * i - j;

5. Using C++ convert the following into two’s complement big-endian binary:
where unsigned char u, a=0x85, b=0x96, c=02; signed char s, w=0x80, x=0x96, y=0, z=0x15;
For addition and subtraction indicate if overflow and/or carry has occurred.
Show work on a seperate piece of paper.

u = ~ a ;		u = - a ;	
u = a & b;		u = a | b & c;	
u = a ^ b;		u = a + b;	
u = a ^ ’A’;		u = a + ’A’;	
u = a - b;		u = a * b;	
u = a << 2;		u = a >> c;	
u = a * b;		u = a % b;	
u = a / b;		u = - a;	
s = - w ;		s = - z ^ ~ x;	
s = w & x;		s = w ^ x;	
s = w + x;		s = w - x;	
s = x << 2;		s = x >> 2;	