CS 33 Mid Term Exam

UID :_____

Name:_____

Fill in your answers below the questions. If you need more space, you may use the back of the page but be sure to indicate on the front that you are doing so. Be sure to label multi-part answers. All answers are worth 10 points each.

1. Problem # 2.71, CSPP page 124. Topic: type conversion

A. The return expression yields an unsigned result. The specification says that each packed byte is intended to be signed.

B. Either a cast is required or setting the expression into a char and returning that instead would work.

```
char a = ( word >> ( bytenum << 3 ) ) & 0xff ;
return (int) a ;
```

or

```
int left = word << ((3-bytenum) << 3);
return left >> 24;
```

2. Problem # 2.76, CSPP page 126. Topic: multiply by a constant using shift and add

A.	k = 17	(x << 4) + x	16*x+x
B.	k = -7	x -(x << 3)	x-8*x
C.	k = 60	(x << 6) - (x << 2)	64*x-4*x
D.	k = -112	(x << 4) - (x << 7)	16*x-128*x

3. Problem # 2.81, CSPP page 126-127. Topic: Boolean operations Part A

A. (x<y) == (-x>-y). No, Let x = $TMin_{32}$, y = 0.

B. ((x+y)<<4) + y-x == 17*y+15*x. Yes, from the ring properties of two's complement arithmetic.

C. ~x+~y+1 == ~(x+y). Yes,
~x+~y+1 = (-x-1)+(-y-1)+1 = -(x+y)-1 = ~(x+y).

D. (ux-uy) == -(unsigned)(y-x). Yes. Due to the isomorphism between two's complement and unsigned arithmetic.

E. ((x >> 2) << 2) <= x. Yes. Right shift rounds toward minus infinity.

4. Problem # 2.87, CSPP page 129. Topic: Floating point decoding

Format A		**Format B**	
Bits	Value	Bits	Value
1 01111 001	-9/8	1 0111 0010	-9/8
0 10110 011	176	0 1110 0110	176
1 00111 010	$-5/2^{10}$	1 0000 0100	$-5/2^{10}$
0 00000 111	$7/2^{17}$	0 0000 1110	$7/2^{10}$
1 11100 000	-2^{13}	1 1110 1111	-248
0 10111 100	384	0 1110 1111	infinity due roundoff

5. Problem #s 2.92-2.95, CSPP page 132-133. Topic: Bit level float point

For these 5 mini problems,simply write the missing line in the function below as your answer. You do not need to worry about the last paragraph in the problem: what to do in abnormal cases and test all possible values.

The function is:

```
unsigned do_the_problem(unsigned f)
   {
   /* Decompose bit representation into parts */
   unsigned sign = f>>31;
   unsigned exp  = f>>23 & 0xFF;
   unsigned frac = f & 0x7FFFFF;

*** your one line statement would go here, just put it in the answer line ***

   /* Reassemble bits */
   return (sign << 31) | (exp << 23) | frac;
   }
```

Prob	Answer
2.92	sign = 0 ;
2.93	exp = exp+1 ;
2.94	exp = exp-1 ;
2.95	**problem misstated**

6. Problem # 3.58, CSPP page 297-298. Topic: Interpreting assembly code

MODE_A	result = *p1 ; *p1 = *p2 ; break ;
MODE_B	result = *p2 ; *p2 = *p1+*p2 ; break ;
MODE_C	*p2 = 15; result = *p1 ; break ;
MODE_D	*p2 = *p1 ;
MODE_E	result = 17 ; break ;
default	result = -1 ;

7. Problem # 3.60, CSPP page 300. Topic: How arrays are laid out in memory

A. &A[0][0][0]+4*(i*S*T+j*T+k)

B. A[5][9][11]

8. Problem # 3.62, CSPP page 300-301. Topic: How arrays are laid out in memory. *** Do not do part C. ***

A. M = 13

B. I is %edi, j is %ecx

9. Problem # 3.65, CSPP page 304. Topic: How structures containing arrays are laid out in memory

the offset to str1.y is 92 but there could be padding. with padding A*B*2 could be 90 which would require padding

the offset to str2.u minus the offset to str2.t minus 4 would be the sizeof s. but there could be padding so B is either 9 or 10.

if it is 9 there would be padding in both and with A=5 and B=9, A*B*2 = 90 and there would be padding in str1. so A=5, B=9.

10. Problem # 3.67, CSPP page 305. Topic: Unions and structures. Use IA32 pointer sizes.

A. e1.p offset 0; e1.y offset 4; e2.x offset 0; e2.next offset4

B. 8 bytes

C. up -> e2.next -> e1.y = *(up -> e2.next -> e1.p) - up -> e2.x