Full Name:_____

EECS 213 Fall 2007

Midterm Exam

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name on the front.
- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.
- The exam has a maximum score of 55 points.
- The problems are of varying difficulty. The point value of each problem is indicated. Pile up the easy points quickly and then come back to the harder problems.
- This exam is CLOSED BOOK. You may use one sheet of notes (both sides). Good luck!

1 (6):	
2 (12):	
3 (14):	
4 (7):	
5 (10):	
6 (8):	
TOTAL (55):	

Problem 1. (6 points):

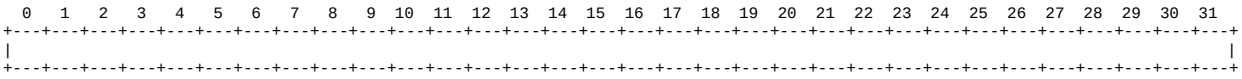
Consider the following datatype definitions on an IA32 (x86) machine.

```
typedef struct {
  char  c;
  double *p;
  int i;
  double d;
  short s;
} struct1;
```

```
typedef union {
  char  c;
  double *p;
  int i;
  double d;
  short s;
} union1;
```

A. Using the template below (allowing a maximum of 32 bytes), indicate the allocation of data for a structure of type `struct1`. Mark off and label the areas for each individual element (there are 5 of them). Cross hatch the parts that are allocated, but not used (to satisfy alignment).

Assume the alignment rules discussed in lecture: data types of size x must be aligned on x-byte boundaries. **Clearly indicate the right hand boundary of the data structure with a vertical line**.



B. How many bytes are allocated for an object of type `struct1`?

C. What alignment is required for an object of type `struct1`? (If an object must be aligned on an x-byte boundary, then your answer should be x.)

D. If we define the fields of `struct1` in a different order, we can reduce the number of bytes wasted by each variable of type `struct1`. What is the number of **unused, allocated** bytes in the best case?

E. How many bytes are allocated for an object of type `union1`?

F. What alignment is required for an object of type `union1`? (If an object must be aligned on an x-byte boundary, then your answer should be x.)

Problem 2. (12 points):

In the following questions assume the variables `a` and `b` are signed integers and that the machine uses two's complement representation. Also assume that `MAX_INT` is the maximum integer, `MIN_INT` is the minimum integer, and `W` is one less than the word length (e.g., `W` = 31 for 32-bit integers).

Match each of the descriptions on the left with a line of code on the right (write in the letter). You will be given 2 points for each correct match.

1. One's complement of `a`

_____

2. `a`.

_____

3. `a & b`.

_____

4. `a * 7`.

_____

5. `a / 4` .

_____

6. `(a < 0) ? 1 : -1` .

_____

a. `~(~a | (b ^ (MIN_INT + MAX_INT)))`

b. `((a ^ b) & ~b) | (~(a ^ b) & b)`

c. `1 + (a << 3) + ~a`

d. `(a << 4) + (a << 2) + (a << 1)`

e. `((a < 0) ? (a + 3) : a) >> 2`

f. `a ^ (MIN_INT + MAX_INT)`

g. `~((a | (~a + 1)) >> W) & 1`

h. `~((a >> W) << 1)`

i. `a >> 2`

Problem 3. (14 points):

Consider the following 8-bit floating point representation based on the IEEE floating point format:

- There is a sign bit in the most significant bit.
- The next 3 bits are the exponent. The exponent bias is $2^{3-1} - 1 = 3$.
- The last 4 bits are the fraction.
- The representation encodes numbers of the form: $V = (-1)^s \times M \times 2^E$, where M is the significand and E is the biased exponent.

The rules are like those in the IEEE standard(normalized, denormalized, representation of 0, infinity, and NAN). FILL in the table below. Here are the instructions for each field:

- **Binary:** The 8 bit binary representation.
- **M:** The value of the significand. This should be a number of the form x or $\frac{x}{y}$, where x is an integer, and y is an integral power of 2. Examples include 0, $\frac{3}{4}$.
- **E:** The integer value of the exponent.
- **Value:**The numeric value represented.

  Note: you need not fill in entries marked with "—".

Description	Binary	M	E	Value
Minus zero				-0.0
—	`0 100 0101`			
Smallest denormalized (negative)				
Largest normalized (positive)				
One				1.0
—				5.5
Positive infinity		—	—	$+\infty$

Problem 4. (7 points):

Match each of the assembler routines on the left with the equivalent C function on the right.

```
foo1:
     pushl %ebp
     movl %esp,%ebp
     movl 8(%ebp),%eax
     sall $4,%eax
     subl 8(%ebp),%eax
     movl %ebp,%esp
     popl %ebp
     ret

foo2:
     pushl %ebp
     movl %esp,%ebp
     movl 8(%ebp),%eax
     testl %eax,%eax
     jge .L4
     addl $15,%eax
.L4:
     sarl $4,%eax
     movl %ebp,%esp
     popl %ebp
     ret

foo3:
     pushl %ebp
     movl %esp,%ebp
     movl 8(%ebp),%eax
     shrl $31,%eax
     movl %ebp,%esp
     popl %ebp
     ret
```

```
int choice1(int x)
{
    return (x < 0);
}

int choice2(int x)
{
    return (x << 31) & 1;
}

int choice3(int x)
{
    return 15 * x;
}

int choice4(int x)
{
    return (x + 15) /4
}

int choice5(int x)
{
    return x / 16;
}

int choice6(int x)
{
    return (x >> 31);
}
```

Fill in your answers here:

foo1 corresponds to choice _____.

foo2 corresponds to choice _____.

foo3 corresponds to choice _____.

Problem 5. (10 points):

Consider a **6-bit** two's complement representation. Fill in the empty boxes in the following table:

Number	Decimal Representation	Binary Representation
Zero	0	
n/a	-1	
n/a	5	
n/a	-10	
n/a		01 1010
n/a		10 0110
TMax		
TMin		
TMax+TMax		
TMin+TMin		
TMin+1		
TMin−1		
TMax+1		
−TMax		
−TMin		

Problem 6. (8 points):

Consider the following assembly representation of a function `foo` containing a `for` loop:

```
foo:
  pushl %ebp
  movl %esp,%ebp
  pushl %ebx
  movl 8(%ebp),%ebx
  leal 2(%ebx),%edx
  xorl %ecx,%ecx
  cmpl %ebx,%ecx
  jge .L4
.L6:
  leal 5(%ecx,%edx),%edx
  leal 3(%ecx),%eax
  imull %eax,%edx
  incl %ecx
  cmpl %ebx,%ecx
  jl .L6
.L4:
  movl %edx,%eax
  popl %ebx
  movl %ebp,%esp
  popl %ebp
  ret
```

Fill in the blanks to provide the functionality of the loop:

```
int foo(int a)
{
    int i;
    int result = _____________;

    for( _________; _________; i++ ) {

        ____________________;

        ____________________;

    }
    return result;
}
```