Andrew login ID: _____

Full Name: _____

Recitation Section: _____

CS 15-213, Spring 2008

Exam 1

Tue. February 26, 2008

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name, Andrew login ID, and recitation section (A–H) 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 70 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 OPEN BOOK. You may use any books or notes you like. No calculators or other electronic devices are allowed.
- Good luck!

1 (8):	
2 (8):	
3 (10):	
4 (9):	
5 (8):	
6 (8):	
7 (11):	
8 (8):	
TOTAL (70):	

Problem 1. (8 points):

For this problem, assume the following:

- We are running code on a 6-bit machine using two's complement arithmetic for signed integers.
- `short` integers are encoded using 3 bits.
- Sign extension is performed whenever a `short` is casted to an `int`
- Right shifts of `int`s are arithmetic.

Fill in the empty boxes in the table below. The following definitions are used in the table:

```
int a = -29;
short b = (short)a;
unsigned ua = a;
int x = -21;
short y = (short)x;
unsigned ux = x;
```


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

Expression	Decimal Representation	Binary Representation
—	27	
—		`100100`
`x`		
`y`		
`ux`		
`a >> 2`		
`ua >> 2`		
`b << 1`		
$-$TMin		

Problem 2. (8 points):

Part A

Fill in the blanks in the table below with the number described in the first column of each row. You can give your answers as unexpanded simple arithmetic expressions (such as $15^{213} + 42$); you should not have trouble fitting your answers into the space provided.

Description	Number
`int x=1; float *f = (float *)&x;` What is the value of `*f`?	
`int x=-1; float *f = (float *)&x;` What is the value of `*f`?	
Smallest positive integer that cannot be represented as a 32-bit `float`	

Part B

Assume we are running code on an IA32 machine, which has a 32-bit word size and uses two's complement arithmetic for signed integers. Consider the following definition:

```
int x = foo();
```


Fill in the empty boxes in the table below. For each of the C expressions in the first column, either:

- State that it is true of all argument values, or
- Give an example where it is not true.

Puzzle	True / Counterexample
`x < 0` ⇒ `-x > 0`	
`x ^ ~x < 0`	
`(x ^ (x >> 31)) + 1 > 0`	
`(((!!x) << 31) >> 31) & x == x`	

Problem 3. (10 points):

Consider an 8-bit IEEE floating-point representation with:

- 1 sign bit
- 3 exponent bits (therefore the bias $B = 2^{3-1} - 1 = 3$)
- 4 mantissa bits

A. Fill in the blanks in the following table. Express numerical values as fractions (e.g., $277/512$).

Number	Bit representation
$3/8$	
$-\infty$	
$9/2$	
	`0 010 0101`
	`1 000 1000`
	`0 111 0010`

B. Give the bit representation and numerical value of the largest number representable in this format as a *denormalized* floating-point number.

C. Give the bit representation and numerical value of the largest number representable in this format as a *normalized* floating-point number.

Problem 4. (9 points):

Consider the following x86-64 assembly code:

```
# on entry: %rdi = n, %rsi = A
000000000040056e <bar>:
  40056e:       41 b8 00 00 00 00       mov    $0x0,%r8d
  400574:       41 b9 00 00 00 00       mov    $0x0,%r9d
  40057a:       41 39 f9                cmp    %edi,%r9d
  40057d:       7d 30                   jge    4005af <bar+0x41>
  40057f:       ba 00 00 00 00          mov    $0x0,%edx
  400584:       39 fa                   cmp    %edi,%edx
  400586:       7d 1f                   jge    4005a7 <bar+0x39>
  400588:       49 63 c1                movslq %r9d,%rax
  40058b:       48 8b 0c c6             mov    (%rsi,%rax,8),%rcx
  40058f:       48 63 c2                movslq %edx,%rax
  400592:       8b 04 81                mov    (%rcx,%rax,4),%eax
  400595:       41 0f af c1             imul   %r9d,%eax
  400599:       0f af c2                imul   %edx,%eax
  # Instruction "cltq" is equivalent to "movslq %eax, %rax"
  40059c:       48 98                   cltq
  40059e:       49 01 c0                add    %rax,%r8
  4005a1:       ff c2                   inc    %edx
  4005a3:       39 fa                   cmp    %edi,%edx
  4005a5:       7c e8                   jl     40058f <bar+0x21>
  4005a7:       41 ff c1                inc    %r9d
  4005aa:       41 39 f9                cmp    %edi,%r9d
  4005ad:       7c d0                   jl     40057f <bar+0x11>
  4005af:       4c 89 c0                mov    %r8,%rax
  4005b2:       c3                      retq
```


Fill in BOTH of the blanks below for the corresponding C code.

```
long bar(int n, ________________ A) {         // Fill in type of A
    long sum = 0;
    int i,j;
    for (i = 0; i < n; i++) {
        for(j = 0; j < n; j++)

            sum += _____________________;   // Fill in expression
    }
    return sum;
}
```

Problem 5. (8 points):

Consider the C code below, where `H` and `J` are constants declared with `#define`.

```
int array1[H][J];
int array2[J][H];

int copy_array(int x, int y) {
    array2[y][x] = array1[x][y];

    return 1;
}
```

Suppose the above C code generates the following x86-64 assembly code:


```
# On entry:
#    %edi = x
#    %esi = y
#
copy_array:
    movslq  %edi,%rdi
    movslq  %esi,%rsi
    movq    %rdi, %rax
    leaq    (%rsi,%rsi,8), %rdx
    salq    $5, %rax
    subq    %rdi, %rax
    addq    %rdi, %rdx
    leaq    (%rsi,%rax,2), %rax
    movl    array1(,%rax,4), %eax
    movl    %eax, array2(,%rdx,4)
    movl    $1, %eax
    ret
```

What are the values of `H` and `J`?

`H =`

`J =`

Problem 6. (8 points):

Consider the following data structure declaration:

```
struct node{
  char x;
  int array[2];
  int idx;
  struct node *next;
};
```

Below are given four C functions and four x86-64 code blocks.

```
char *mon(struct node *ptr){
  return &ptr->x;
}
```

```
int tue(struct node *ptr){
  return ptr->array[ptr->idx];
}
```

```
void wed(struct node *ptr){
  ptr->idx = ptr->array[1];
  return;
}
```

```
char thu(struct node *ptr){
  ptr = ptr->next;
  return ptr->x;
}
```

A	movl 8(%rdi), %eax movl %eax, 12(%rdi)
B	movq %rdi, %rax
C	movq 16(%rdi), %rax movsbl (%rax),%eax
D	movslq 12(%rdi),%rax movl 4(%rdi,%rax,4), %eax

In the following table, next to the name of each x86-64 code block, write the name of the C function that it implements.

Code Block	Function Name
A	
B	
C	
D	

Problem 7. (11 points):

The next problem concerns code generated by GCC for a function involving a switch statement. The code uses a jump to index into the jump table:

```
  400518: ff 24 d5 40 06 40 00  jmpq   *0x400640(,%rdx,8)
```

Using GDB, we extract the 8-entry jump table as:

```
0x400640:    0x0000000000400527
0x400648:    0x0000000000400525
0x400650:    0x0000000000400531
0x400658:    0x000000000040051f
0x400660:    0x0000000000400525
0x400668:    0x0000000000400525
0x400670:    0x000000000040052a
0x400678:    0x0000000000400531
```


The following block of disassembled code implements the branches of the switch statement:

```
# on entry: %rdi = a, %rsi = b, %rdx = c
  400510: 31 c0                 xor    %eax,%eax
  400512: 48 83 fa 07           cmp    $0x7,%rdx
  400516: 77 0d                 ja     400525 <_Z4testlll+0x15>
  400518: ff 24 d5 40 06 40 00  jmpq   *0x400640(,%rdx,8)
  40051f: 48 89 f0              mov    %rsi,%rax
  400522: 48 29 f8              sub    %rdi,%rax
  400525: f3 c3                 repz retq # repz is a no-op here
  400527: 48 01 f7              add    %rsi,%rdi
  40052a: 48 89 f8              mov    %rdi,%rax
  40052d: 48 31 f0              xor    %rsi,%rax
  400530: c3                    retq
  400531: 48 8d 46 2a           lea    0x2a(%rsi),%rax
  400535: c3                    retq
```

Fill in the blank portions of C code below to reproduce the function corresponding to this object code. You can assume that the first entry in the jump table is for the case when `c` equals 0.

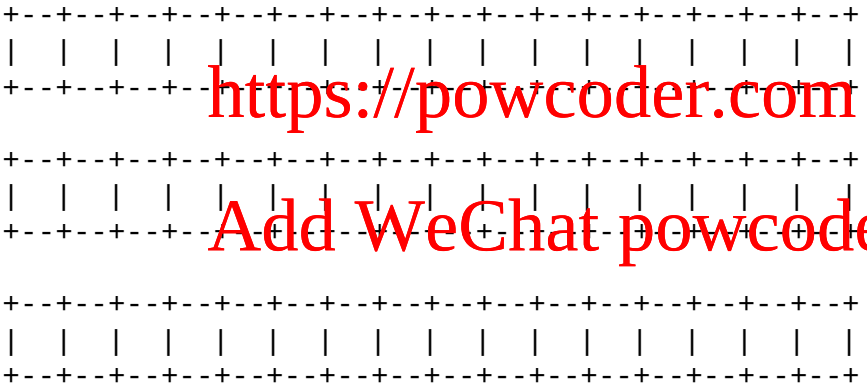
```
long test(long a, long b, long c)
{
  long answer = _____;
  switch(c)
  {
    case ___:
      answer = _____;
      break;
    case ___:
    case ___:
      answer = _____;
      break;
    case ___:
      a = _____;
      /* Fall through */
    case ___:
      answer = _____;
      break;
    default:
      answer = _____;
  }

  return answer;
}
```

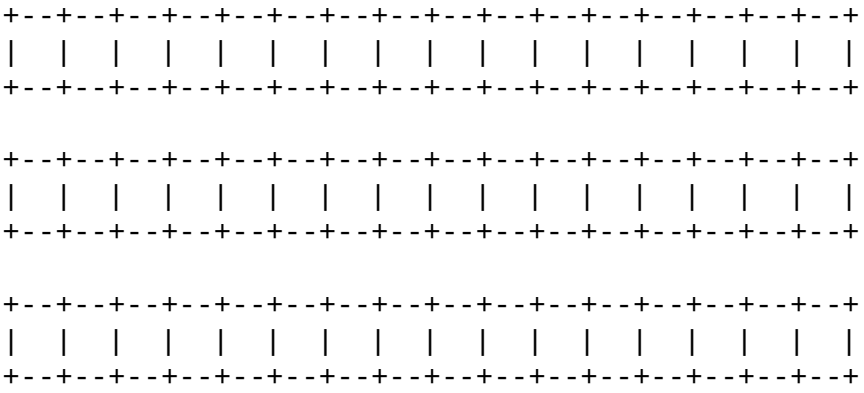
Problem 8. (8 points):

```
struct BOOKLIST {
    char a;
    short US;
    char b;
    short CA;
    char c;
    double EU;
    char d;
    int UK;
} booklist;
```

A. Show how the struct above would appear on a 32 bit Windows machine (primitives of size k are k byte aligned). Label the bytes that belong to the various fields with their names and clearly mark the end of the struct. Use hatch marks to indicate bytes that are allocated in the struct but are not used.



B. Rearrange the above fields in `booklist` to conserve the most space in the memory below. Label the bytes that belong to the various fields with their names and clearly mark the end of the struct. Use hatch marks to indicate bytes that are allocated in the struct but are not used.



C. How many bytes of the struct are wasted in part A?

D. How many bytes of the struct are wasted in part B?