

Andrew login ID:.....

Full Name:.....

CS 15-213, Fall 2006

Exam 1

Wednesday October 4, 2006

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name and Andrew login ID 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 56 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. Calculators are allowed, but no other electronic devices. Good luck!

1 (8):
2 (8):
3 (8):
4 (6):
5 (8):
6 (8):
7 (10):
TOTAL (56):

Problem 1. (8 points):

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 definitions:

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

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
$x < 0 \Rightarrow (x * 2) < 0$	False (TMin)
$x > 0 \Rightarrow (x + 1) > 0$	
$x > 0 \Rightarrow (\sim x + 2) \leq 0$	
$(x \gg 31) == -1 \Rightarrow x < 0U$	
$x < 0 \Rightarrow ((x \wedge x \gg 31) + 1) > 0$	
$((x \gg 31) + 1) == (x >= 0)$	
$x \geq 0 \Rightarrow ((!x - 1) \& x) == x$	
$((int)(ux \gg 31) + \sim 0) == -1$	
$-(x \mid (\sim x + 1)) > 0$	

Problem 2. (8 points):

Consider the following 5-bit floating point representations based on the IEEE floating point format. This format does not have a sign bit – it can only represent nonnegative numbers.

- There are $k = 3$ exponent bits. The exponent bias is 3.
- There are $n = 2$ fraction bits.

Numeric values are encoded as a value of the form $V = M \times 2^E$, where E is exponent after biasing, and M is the significand value. The fraction bits encode the significand value M using either a denormalized (exponent field 0) or a normalized representation (exponent field nonzero).

Below, you are given some decimal values, and your task is to encode them in floating point format. If rounding is necessary, you should use *round-to-even*, as you did in Lab 1 for the `float_i2f` puzzle. In addition, you should give the rounded value of the encoded floating point number. Give these as whole numbers (e.g., 17) or as fractions in reduced form (e.g., $3/4$).

Value	Floating Point Bits	Rounded value
$9/32$	001 00	$1/4$
$7/8$		
$15/16$		
9		
10		

Problem 3. (8 points):

Consider the following C function's x86-64 assembly code:

```
# On entry %edi = n
#
000000004004a8 <foo>:
4004a8:  b8 00 00 00 00      mov     $0x0,%eax
4004ad:  83 ff 01             cmp     $0x1,%edi
4004b0:  7e 1a               jle    4004cc <foo+0x24>
4004b2:  01 f8               add    %edi,%eax
4004b4:  ba 00 00 00 00      mov     $0x0,%edx
4004b9:  39 fa               cmp    %edi,%edx
4004bb:  7d 08               jge    4004c5 <foo+0x1d>
4004bd:  01 d0               add    %edx,%eax
4004bf:  ff c2               inc    %edx
4004c1:  39 fa               cmp    %edi,%edx
4004c3:  7c f8               jl     4004bd <foo+0x15>
4004c5:  ff cf               dec    %edi
4004c7:  83 ff 01             cmp    $0x1,%edi
4004ca:  7f e6               jg     4004b2 <foo+0xa>
4004cc:  f3 c3               repz  retq  # treat repz as a no-op
```

Please fill in the corresponding C code:

```
int foo (int n) {
    int a, i;

    a = 0;
    for (; n > _____; _____) {
        a = a + _____;
        for (i = _____; i < _____; _____)
            a = a + _____;
    }
    return _____;
}
```

Problem 4. (6 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 %esi,%rsi
    movslq %edi,%rdi
    movq   %rsi, %rax
    salq  $7, %rax
    subq  %rsi, %rax
    addq  %rdi, %rax
    leaq  (%rdi,%rdi,2), %rdi
    addq  %rsi, %rdi
    movl  array1(,%rdi,4), %edx
    movl  %edx, array2(,%rax,4)
    movl  $1, %eax
    ret
```

What are the values of H and J?

H =

J =

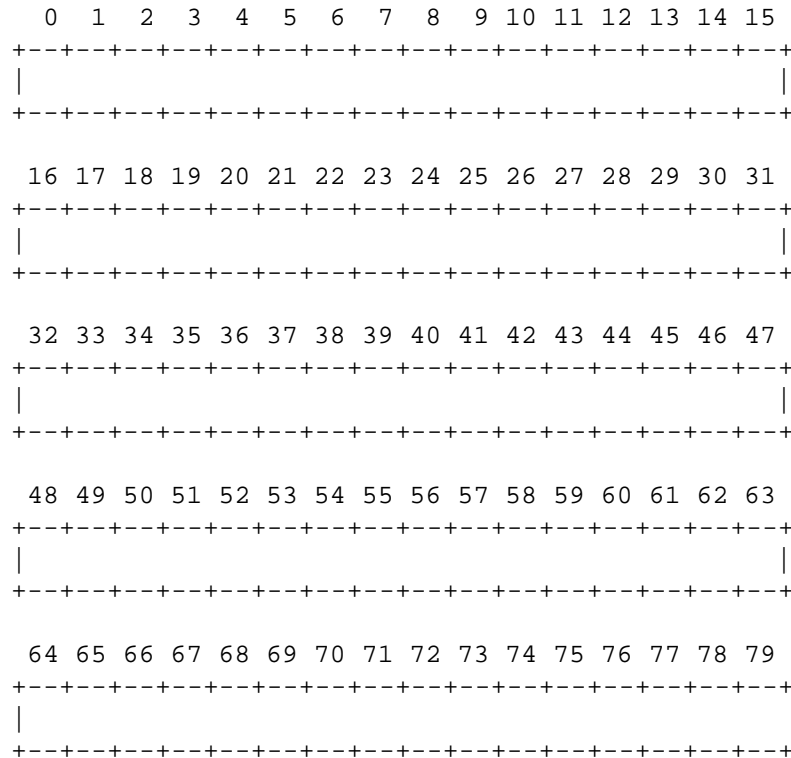
Problem 5. (8 points):

Consider the following C declaration:

```
typedef struct WineNode {
    int vintages[3];
    double cost;
    char z;
    WineNode *next;
    short ages[5];
    int type;
    char a;
} WineNode;
```

A. Using the template below (allowing a maximum of 80 bytes), indicate the allocation of data for the struct WineNode. Mark off and label the areas for each element (arrays may be labeled as a single element). **Cross hatch the parts that are allocated, but not used. Clearly mark the end of the struct. Assume the 64 bit alignment rules and X86-64 data structure sizes discussed in class.**

WineNode:



B. How many bytes of space in WineNode are wasted?_____

C. Now rewrite the WineNode struct in the space provided below **so that the amount of wasted allocated space in WineNode is minimized.**

```
typedef struct WineNode {
```

```
} WineNode;
```

D. Now rewrite the allocation for WineNode as you did before using this new specification.

WineNode:

```
  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                             |
+-----+-----+-----+-----+-----+-----+-----+-----+

 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                             |
+-----+-----+-----+-----+-----+-----+-----+-----+

 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                             |
+-----+-----+-----+-----+-----+-----+-----+-----+

 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                             |
+-----+-----+-----+-----+-----+-----+-----+-----+

 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                             |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

E. How many bytes of space in this new WineNode are wasted?_____

Problem 6. (8 points):

Consider the following data structure declarations:

```
struct node {
    struct data d;
    struct node *next;
};

struct data {
    int x;
    char str[6];
};
```

Below are given four C functions and four x86-64 code blocks. Next to each of the x86-64 code blocks, write the name of the C function that it implements.

```
int alpha(struct node *ptr) {
    return ptr->d.x;
}
```

```
movq    16(%rdi), %rax
addq    $4, %rax
ret
```

```
char *beta(struct node *ptr) {
    ptr = ptr->next;
    return ptr->d.str;
}
```

```
movq    %rdi, %rax
ret
```

```
char gamma (struct node *ptr) {
    return ptr->d.str[4];
}
```

```
movl    (%rdi), %eax
ret
```

```
int *delta (struct node *ptr) {
    struct data *dp =
        (struct data *) ptr;
    return &dp->x;
}
```

```
movsbl  8(%rdi),%eax
ret
```


Reverse Engineering Switch Code

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

```
400476: ff 24 d5 a0 05 40 00  jmpq  *0x4005a0(,%rdx,8)
```

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

```
0x4005a0: 0x0000000000400480 0x0000000000400491
0x4005b0: 0x0000000000400480 0x0000000000400496
0x4005c0: 0x0000000000400480 0x0000000000400489
0x4005d0: 0x0000000000400485 0x0000000000400496
```

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

```
400480: 48 8d 04 3f  lea  (%rdi,%rdi,1),%rax
400484: c3          retq
400485: 48 0f af f7  imul %rdi,%rsi
400489: 48 89 f8     mov  %rdi,%rax
40048c: 48 21 f0     and  %rsi,%rax
40048f: 90          nop
400490: c3          retq
400491: 48 8d 04 37  lea  (%rdi,%rsi,1),%rax
400495: c3          retq
400496: 48 8d 46 ff  lea  0xffffffffffffffff(%rsi),%rax
40049a: c3          retq
```

Problem 7. (10 points):

Fill in the blank portions of the 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 `s` equals 0. Parameters `a`, `b`, and `s` are passed in registers `%rdi`, `%rsi`, and `%rdx`, respectively.

```
long fun(long a, long b, long s)
{
    long result = 0;
    switch (s) {
    case ____:
    case ____:
        result = ____;
        break;
    case ____:
        b = ____;
        /* Fall through */
    case ____:
        result = ____;
        break;
    case ____:
        result = ____;
        break;
    default:
        result = ____;
    }
    return result;
}
```