CS 211: Midterm : 100 points

Instructor: Prof. Santosh Nagarakatte

Full Name Here:
RUID:

Question	Max Points	Points
1	20	
2	20	
3	20	
4	20	
5	20	

Problem 1: C Programming (20 points)

1. (10 points) You are implementing a hash table with open chaining where each node is of the following type.

```
struct node{
   int key;
   int value;
   struct node * next;
};
```

Given that the hash table is implemented as an array of pointers to hash table nodes, implement the following hash_search function to search a key in the hash table. If the key is found, the search function returns the value associated with the key and returns -1 otherwise. The value MAX_ENTRIES is the number of buckets in the hash table. Your code should carefully handle all corner cases, should compile, and should not experience segmentation faults on any input.

```
struct node * hash_table[MAX_ENTRIES];

int hash_search(int key){
```

2. (10 points) In the following code, we have omitted the definitions of constants M and N:

```
#define M  /* Mystery number 1 */
#define N /* Mystery number 2 */

unsigned int arith(unsigned int x, unsigned int y){
    unsigned int result = 0;
    result = x * M + y/N;
    return result;
}
```

We compiled this code for particular values of M and N. The compiler optimized the multiplication and division. The following is a translation of the generated machine code back into C:

```
unsigned int optarith(unsigned int x, unsigned int y){
   unsigned int t = x;
   x <<= 3;
   x -= t;
   y >>= 4;
   return x + y;
}
```

What are the values of M and N?

Problem 2: Data Representation (20 points)

1. (10 points) Consider a 8-bit floating-point representation based on the IEEE floating point format with one sign bit, three exponent bits, and four bits for the fractional part.

   - (1 point) What is the bias in this representation?
   - (6 points) Represent 0.1 (decimal value) in this representation. You have to get everything correct to get points. Show work.
     - sign bit:
     - exponent bits:
     - mantissa bits:
   - (3 points) What is the largest normalized value in this representation?
     - sign bit:
     - exponent bits:
     - mantissa bits:

2. (6 points) Consider you are designing **6-bit signed integer** representation.

   - Represent -25 in two's complement representation:
   - Represent 27 in two's complement representation:

3. (4 points) You are given a K-bit binary number. You take the one's complement of it and add it to the original number. What is the value of the number when interpreted as an unsigned integer? Why?

Problem 4: Assembly (20 points)

1. (8 points) You are given the following information. A function with prototype

   ```
   void decode1(int *xp, int *yp, int *zp);
   ```

   is compiled into assembly code, yielding the following:

   ```
   xp is in %edi, yp in %esi, zp in %edx

   decode1:
      movl (%edi), %ebx
      movl (%esi), %ecx
      movl (%edx), %eax
      movl %ebx, (%esi)
      movl %ecx, (%edx)
      movl %eax, (%edi)
      ret
   ```

   Parameters xp, yp, and zp are stored in %edi, %esi, and %edx respectively.

   Write C code for decode1 that will have an effect equivalent to the assembly code shown.

2. (12 points) Assume the following values are stored at the indicated memory addresses and registers.

Address	Value
0x100	0xFF
0x108	0xAB
0x110	0x13
0x118	0x11

Register	Value
%eax	0x100
%ecx	0x1
%edx	0x3

Fill in the following table showing the effects of the following instructions, in terms of both the register or memory location that will be updated and the resulting value. If the destination is a memory location, provide the effective address.

Instruction	Destination	Value
`addl %ecx, (%eax)`		
`subl %edx, 8(%eax)`		
`imull $16, (%eax, %edx, 8)`		
`incl 16(%eax)`		
`decl %ecx`		
`subl %edx, %eax`		

Here addl, subl, imull, incll, decl are addition, subtraction, integer multiplication, increment, and decrement operations respectively. All operations are performed on integers.

Problem 5: Bomblab (20 points)

As with the bomblab you have to devise the input to this program. There are multiple inputs that solve this phase named *foo*. Each input takes two integers. **Identify all the inputs that would defuse this phase**. The function *explode_bomb* has the same behavior as in bomblab. The function *scanf* is the other function used in this program. This program can be diffused without using gdb.

```
.LC0:
        .string "%d"
        .text
.globl foo
        .type   foo, @function
foo:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $40, %esp
        leal    -12(%ebp), %eax
        movl    %eax, 4(%esp)
        movl    $.LC0, (%esp)
        call    scanf
        movl    -12(%ebp), %eax
        leal    0x1(%eax, %eax), %eax
        cmpl    $3, %eax
        je      .L3
        cmpl    $5, %eax
        jne     .L7
        jmp     .L8

.L3:
        leal    -16(%ebp), %eax
        movl    %eax, 4(%esp)
        movl    $.LC0, (%esp)
        call    scanf
        cmpl    $17, -16(%ebp)
        je      .L6
        call    explode_bomb
.L8:
        leal    -16(%ebp), %eax
        movl    %eax, 4(%esp)
        movl    $.LC0, (%esp)
        call    scanf
        movl    -16(%ebp), %ebx
        leal    0x1(%ebx), %ebx
        cmpl    $19, %ebx
        je      .L6
.L7:
        call    explode_bomb
.L6:
        leave
        ret
```