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: Data Representation (20 points)

1. (16 points) Consider a 7-bit floating-point representation based on the IEEE floating point format with one sign bit, three exponent bits, and three fraction bits.

   - (1 point) What is the bias in this representation?
   - (3 points) Provide the following information for the largest normal positive number in this representation?
     - sign bit:
     - exponent bits:
     - mantissa bits:
   - (3 points) Provide the following information for the largest denormal positive number in this representation?
     - sign bit:
     - exponent bits:
     - mantissa bits:
   - (3 points) Provide the following information for the smallest normal positive number in this representation?
     - sign bit:
     - exponent bits:
     - mantissa bits:
   - (3 points) Provide the following information for the smallest denormal positive number in this representation?
     - sign bit:
     - exponent bits:
     - mantissa bits:
   - (3 points) Provide the following information for the number One in this representation?
     - sign bit:
     - exponent bits:
     - mantissa bits:

2. (4 points) Consider you are designing **7-bit signed integer** representation. What are the maximum and minimum values you can represent?

   - Max value in two's complement representation:
   - Min value in two's complement representation:
   - Max value in one's complement representation:
   - Min value in one's complement representation:

Problem 2: C Programming (20 points)

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

   ```
   struct node {
       int id;          // Represents hash key
       int data;        // Data of an item;
       struct node * link;
   };
   ```

   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.

   Use (`key` + 42) modulo `MAX_ENTRIES` as the hash function.

   ```
   struct node * hash_table [MAX_ENTRIES];

   int hash_search (int key) {
   ```

Problem 3: C Programming: Mystery Numbers (20 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 = 3 * x;
   x <<= 4;
   x -= 3 * t;
   y >>= 2;
   y = y / 7;
   return x + y;
}
```

What are the values of `M` and `N`? (show work)

Problem 4: Bomblab (20 points)

The following is the binary given for your bomblab assignment. However, we have simplified the binary so that you can solve this assignment without gdb. Your goal is to find all inputs that diffuse the bomb. Each input consists of 3 items: an integer, a character, a character.

```
.LC0
         .string  "%d %c %c"
         .text
.globl phase1
         .type    phase1, @function
phase1:
         pushl    %ebp
         movl           %esp, %ebp
         subl     $56, %esp
         leal     -14(%ebp), %eax
         movl     %eax, 16(%esp)
         leal     -13(%ebp), %eax
         movl     %eax, 12(%esp)
         leal     -12(%ebp), %eax
         movl     %eax, 8(%esp)
         movl     $.LC0, 4(%esp)
         movl     8(%ebp), %eax
         movl     %eax, (%esp)
         call     sscanf
         movl     -12(%ebp), %edx
         leal     3(%edx), %eax
         cmpl     $9, %eax
         je       .L9
.L2:
         leal     0xf(,%edx, 4), %eax
         cmpl     $23, %eax
         je       .L9

.L5:
         leal     -0x2(%eax, %edx, 2), %eax
         cmpl     $97, %eax
         je       .L9
.L8:
         call     explode_bomb
.L9:
         leave
         ret
```

1. (5 points) How many inputs solve the bomb?

2. (15 points) List all inputs that solve the bomb. You are required to list only inputs that need a distinct integer input.

Problem 5: Assembly Programming (20 points)

1. Write C code for the following assembly code.

```
.globl my_func
         .typ      my_func, @function
my_func:
         pushl     %ebp
         movl      %esp, %ebp
         movl      8(%ebp), %edx
         movl      $1, %eax
         testl     %edx, %edx
         jle       .L4
         movl      $1, %eax
.L5:
         addl      $4, %eax
         imull     %edx, %eax
         subl      $1, %edx
         testl     %edx, %edx
         jg        .L5
.L4:
         popl      %ebp
         ret
```

2. (10 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	0x11
%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. If the destination is a register, provide the register name.

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

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