1.
Floating Point
Convert the 32-bit floating point number 0x44361000 to decimal.

(Source: http://sandbox.mc.edu/~bennet/cs110/flt/ftod.html)

2.
Fill in the Blanks:

_____ linking can suffer from issues such as code duplication, whereas _____ linking may take longer during runtime.

x86-64 is a (RISC/CISC) architecture, and MIPS is a (RISC/CISC) architecture.

A _____ is an array of page table entries (PTEs) that maps virtual pages to physical pages.

3.
Consider the following union and struct:

```
struct Galor {
    int first;
    float second;
    char third;

    union Hello {
        struct Hi {
            int number;
            float frac;
        };
        char name[10];
    };
};
```

Say we are debugging an application in execution using gdb on a 64-bit, little-endian architecture. The application has a variable called Sword, defined as:
`struct Galor Sword[2][2];`

Using gdb we find the following information at a particular stage in the application:

```
(gdb) p &Sword
$1 = (struct Galor (*)[2][2]) 0x7fffffffdff0
(gdb) x/96xb 0x7fffffffdff0
0x7fffffffdff0: 0x6b    0x72    0x00    0x00    0xec    0x51    0x05    0x42
0x7fffffffdff8: 0x3f    0x00    0x00    0x00    0x5a    0x61    0x6d    0x61
0x7fffffffe000: 0x7a    0x65    0x6e    0x74    0x61    0x00    0x00    0x00
0x7fffffffe008: 0x15    0x16    0x05    0x00    0xf5    0x19    0xd2    0x42
0x7fffffffe010: 0x2f    0x00    0x00    0x00    0x57    0x6f    0x6f    0x6c
0x7fffffffe018: 0x6f    0x6f    0x00    0x00    0x00    0x00    0x00    0x00
0x7fffffffe020: 0xe7    0x66    0xff    0xff    0x5c    0x2a    0x09    0x50
0x7fffffffe028: 0x32    0x00    0x00    0x00    0x43    0x53    0x33    0x33
0x7fffffffe030: 0x00    0x00    0xc8    0x43    0x00    0x00    0x00    0x00
0x7fffffffe038: 0x35    0x00    0x00    0x00    0x56    0x03    0x56    0xc3
0x7fffffffe040: 0x61    0xe1    0xff    0xff    0x44    0x72    0x65    0x64
0x7fffffffe048: 0x6e    0x61    0x77    0x00    0x00    0x00    0x00    0x00
```

What is the value of
`Sword[1][0].frac`
`Sword[1][0].name`

At this particular stage of the application?

4.
Translate the x86 instructions into MIPS and vice versa:

a.
```
lea 0x4(%rdi,%rsi),%rax
```

b.
```
mov %rdx,(%rsp,%rsi,8)
```

c.
```
add $t1, $t0, $t0
add $t1, $t1, $t1
add $t3, $t2, $t1
lw $t3, 128($t3)
add $t4, $t4, $t3
```

5.
Is there a problem with the following code?
If yes, what is it? How can we fix the problem if there is one?

```
double* input = (double*) malloc (sizeof(double)*dnum);
double sum = 0;
int i;
for(i=0;i<dnum;i++){
      input[i] = i+1;
}

#pragma omp parallel for schedule(static)
for(i=0;i<dnum;i++)
{
      double* tmpsum = input+i;
      sum += *tmpsum;
}
```

6.
We have a function that we are interested in:

```
int Toronto(char* game) {
      int curr_game = atoi(game);

      return Raptors(curr_game, 0);
}
```

We only know that the function `Raptors` has the following declaration:
`int Raptors(int game, int wins)`

While debugging, we notice the following output:

```
(gdb) disas Raptors
Dump of assembler code for function Raptors:
   0x000000000040068d <+0>:     push   %rbp
   0x000000000040068e <+1>:     mov    %rsp,%rbp
   0x0000000000400691 <+4>:     sub    $0x10,%rsp
   0x0000000000400695 <+8>:     mov    %edi,-0x4(%rbp)
   0x0000000000400698 <+11>:    mov    %esi,-0x8(%rbp)
   0x000000000040069b <+14>:    mov    -0x4(%rbp),%eax
   0x000000000040069e <+17>:    sub    -0x8(%rbp),%eax
   0x00000000004006a1 <+20>:    test   %eax,%eax
   0x00000000004006a3 <+22>:    js     0x4006bc <Raptors+47>
   0x00000000004006a5 <+24>:    mov    -0x8(%rbp),%eax
   0x00000000004006a8 <+27>:    lea    0x1(%rax),%edx
   0x00000000004006ab <+30>:    mov    -0x4(%rbp),%eax
   0x00000000004006ae <+33>:    sub    $0x1,%eax
   0x00000000004006b1 <+36>:    mov    %edx,%esi
   0x00000000004006b3 <+38>:    mov    %eax,%edi
   0x00000000004006b5 <+40>:    callq  0x40068d <Raptors>
   0x00000000004006ba <+45>:    jmp    0x4006ce <Raptors+65>
   0x00000000004006bc <+47>:    cmpl   $0x4,-0x8(%rbp)
   0x00000000004006c0 <+51>:    jne    0x4006c9 <Raptors+60>
   0x00000000004006c2 <+53>:    mov    $0x1,%eax
   0x00000000004006c7 <+58>:    jmp    0x4006ce <Raptors+65>
   0x00000000004006c9 <+60>:    mov    $0x0,%eax
   0x00000000004006ce <+65>:    leaveq
   0x00000000004006cf <+66>:    retq
End of assembler dump.
```

What should be the input into the function `Toronto`, in order to get **a return value of 1**?

7.
Say there was a function called `Warriors` in the Attack Lab, with the following C representation:

```
int Warriors(float* game) {

      float fourth = *(game+3);
      if (fourth == 68.75)
            return 1;

      return 0;
}
```

The function is at memory location `0x40178a`.
You need to execute the code for `Warriors` so that the function returns 1.
What should your input string be?

Your string is inputted using the same `getbuf` function as the Attack Lab, with a **24 byte buffer**.
The buffer begins at memory address `0x400680`.

You can assume that the **stack positions are consistent** from one run to the next, and that the section of memory holding the stack **is executable**.

`movq` S, D

Source S	Destination D							
	%rax	%rcx	%rdx	%rbx	%rsp	%rbp	%rsi	%rdi
%rax	48 89 c0	48 89 c1	48 89 c2	48 89 c3	48 89 c4	48 89 c5	48 89 c6	48 89 c7
%rcx	48 89 c8	48 89 c9	48 89 ca	48 89 cb	48 89 cc	48 89 cd	48 89 ce	48 89 cf
%rdx	48 89 d0	48 89 d1	48 89 d2	48 89 d3	48 89 d4	48 89 d5	48 89 d6	48 89 d7
%rbx	48 89 d8	48 89 d9	48 89 da	48 89 db	48 89 dc	48 89 dd	48 89 de	48 89 df
%rsp	48 89 e0	48 89 e1	48 89 e2	48 89 e3	48 89 e4	48 89 e5	48 89 e6	48 89 e7
%rbp	48 89 e8	48 89 e9	48 89 ea	48 89 eb	48 89 ec	48 89 ed	48 89 ee	48 89 ef
%rsi	48 89 f0	48 89 f1	48 89 f2	48 89 f3	48 89 f4	48 89 f5	48 89 f6	48 89 f7
%rdi	48 89 f8	48 89 f9	48 89 fa	48 89 fb	48 89 fc	48 89 fd	48 89 fe	48 89 ff

Operation	Register R							
	%rax	%rcx	%rdx	%rbx	%rsp	%rbp	%rsi	%rdi
popq R	58	59	5a	5b	5c	5d	5e	5f