1. What is the value of y after both of the following operations?

```
x = x ^ (~y);
y = y ^ x;
```

2. Given the following declarations, do the statements below always evaluate to true?

```
int x = foo();
int y = bar();
unsigned ux = cookie();
```

a.
```
x > ux  ====> (~x+1)  < 0
```

b.
```
ux - 2 >= -2 ====> ux <= 1
```

c.
```
(x^y)^x == (x+y)^((x+y)^y)
```

d.
```
(x < 0) && (y < 0) == (x + y) < 0
```

3.
```
char** apple[5][9];
char* banana[1][9];
char strawberry[4][2];
```

How many bytes of space would these declarations require?

4. Consider the following struct:

```
typedef struct {
      char first;
      int second;
      short third;
} stuff;
```

Say we are debugging an application in execution using gdb on a 64-bit, little-endian architecture. The application has a variable called array - defined as:

```
stuff array[2][2];
```

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

```
(gdb) p &array
$1 = (stuff (*)[2][2]) 0x7fffffffe020
```

And:

```
(gdb) x/48xb 0x7fffffffe020
0x7fffffffe020: 0x61    0x00    0x00    0x00    0x08    0x00    0x00    0x00
0x7fffffffe028: 0x02    0x00    0x00    0x00    0x62    0x00    0x00    0x00
0x7fffffffe030: 0x64    0x00    0x00    0x00    0x04    0x00    0x00    0x00
0x7fffffffe038: 0x63    0x04    0x40    0x00    0xed    0x03    0x00    0x00
0x7fffffffe040: 0xc8    0x00    0xff    0xff    0x64    0x7f    0x00    0x00
0x7fffffffe048: 0x17    0xa6    0x00    0x00    0xe1    0x00    0x00    0x00
```

What is the value of
`array[1][0].second`

At this particular stage of the application?
i.e. what would be returned from the statement:

```
printf("%d\n", array[1][0].second);
```

5. The following is part of the result of the command '`objdump -d`' on an executable

```
00000000004006dd <IronMan>:
  4006dd:       55                      push   %rbp
  4006de:       48 89 e5                mov    %rsp,%rbp
  4006e1:       89 7d ec                mov    %edi,-0x14(%rbp)
  4006e4:       8b 45 ec                mov    -0x14(%rbp),%eax
  4006e7:       c1 e0 04                shl    $0x4,%eax
  4006ea:       89 45 fc                mov    %eax,-0x4(%rbp)
  4006ed:       8b 45 fc                mov    -0x4(%rbp),%eax
  4006f0:       5d                      pop    %rbp
  4006f1:       c3                      retq
```

Say the declaration for the function IronMan was:
`int IronMan(int scraps);`

Given that the integer 23 was passed into the function, what is the return value?

6. The following is a continuation from the previous problem.

```
0000000000400721 <Hulk>:
  400721:       55                      push   %rbp
  400722:       48 89 e5                mov    %rsp,%rbp
  400725:       48 83 ec 20             sub    $0x20,%rsp
  400729:       48 89 7d e8             mov    %rdi,-0x18(%rbp)
  40072d:       48 8b 45 e8             mov    -0x18(%rbp),%rax
  400731:       48 89 c7                mov    %rax,%rdi
  400734:       e8 27 fe ff ff          callq  400560 <atoi@plt>
  400739:       89 45 fc                mov    %eax,-0x4(%rbp)
  40073c:       8b 45 fc                mov    -0x4(%rbp),%eax
  40073f:       89 c7                   mov    %eax,%edi
  400741:       e8 97 ff ff ff          callq  4006dd <IronMan>
  400746:       89 45 f8                mov    %eax,-0x8(%rbp)
  400749:       81 7d f8 8f 01 00 00    cmpl   $0x18f,-0x8(%rbp)
  400750:       7e 10                   jle    400762 <Hulk+0x41>
  400752:       81 7d f8 f4 01 00 00    cmpl   $0x1f4,-0x8(%rbp)
  400759:       7f 07                   jg     400762 <Hulk+0x41>
  40075b:       b8 01 00 00 00          mov    $0x1,%eax
  400760:       eb 05                   jmp    400767 <Hulk+0x46>
  400762:       b8 00 00 00 00          mov    $0x0,%eax
  400767:       c9                      leaveq
  400768:       c3                      retq
```

Given that the function returns 1, what do we know about the value of %edi right before instruction 0x400741 is executed?

7. What is the value of the following 8-bit, tiny floating point number? Note that the exponent field is 4 bits, and the fractional field is 3.

   01100000