

1. **Lost at C? (12 points):** The following problem assumes the following declarations:

```
int x = rand();
float f = foo(); // f is not NaN
unsigned ux = rand();
```

For the following C expressions, circle either Y or N (but not both). If you circle the right answer, you get +2 points. If you circle the wrong answer, you get -1 point. If you do not circle anything, you get 0 points. So do not just guess wildly.

			Always True?	
a.	$x > 0 \Rightarrow ((x < 4) > 5) > 0$	1 Would not hold	Y	<input checked="" type="radio"/> N
b.	$f > 0 \Rightarrow ((f < 4) > 5) > 0$	1.0 Would not hold	Y	<input checked="" type="radio"/> N
c.	$(x >> 20) == (\sim(x >> 20) + 1) \Rightarrow x == (\text{int})(\text{float}) x$		<input checked="" type="radio"/> Y	<input type="radio"/> N
Case 1: Left is true if $x >> 20 == 0$ (int) then Right: Int = Int				
Case 2: Left POS != NEG then Right: Decimal != nondecimal				
d.	$x \leq 0, f \leq 0 \Rightarrow x * f \leq 0$		<input checked="" type="radio"/> Y	<input type="radio"/> N
e.	$x > ux \Rightarrow (\sim x + 1) < 0$	Left side: False no matter what Right side: Could be true if x POS	Y	<input checked="" type="radio"/> N
f.	$ux - 2 \geq -2 \Rightarrow ux \leq 1$		<input checked="" type="radio"/> Y	<input type="radio"/> N

Note that " \Rightarrow " represents an *implication*. $A \Rightarrow B$ means that you assume A is true, and your answer should indicate whether B should be implied by A – i.e. given that A is true, is B always true?

7. **Magic 8 Ball says "Success is not likely" (10 points):** You are debugging an application in execution using gdb on a 32-bit (i.e. pointers use 32 bits), little-endian architecture. The application has a variable called *magic8ball* - defined as

```
char magic8ball[8][8][8];
```

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

```
(gdb) p &magic8ball
$1 = (char (*)[8][8][8]) 0xffffd448
```

And:

```
(gdb) x/256x 0xffffd428
0xffffd428: 0x6279614d 0x00a50065 0x6e6f7257 0x08040067
0xffffd438: 0x65727553 0x00000000 0x656b694c 0x0000796c
0xffffd448: 0x6576654e 0x00000072 0x656b694c 0x0000796c
0xffffd458: 0x0068614e 0x00000000 0x00006f4e 0x00000000
0xffffd468: 0x00736559 0x00000000 0x0068614e 0x00000000
0xffffd478: 0x6279614d 0x00a50065 0x6e6f7257 0x08040067
0xffffd488: 0x6279614d 0x00a50065 0x6576654e 0x00000072
0xffffd498: 0x68676952 0x08040074 0x6e6f7257 0x08040067
0xffffd4a8: 0x6576654e 0x00000072 0x6279614d 0x00a50065
0xffffd4b8: 0x00006f4e 0x00000000 0x68616559 0x00000000
0xffffd4c8: 0x656b694c 0x0000796c 0x0068614e 0x00000000
0xffffd4d8: 0x0068614e 0x00000000 0x00736559 0x00000000
0xffffd4e8: 0x656b694c 0x0000796c 0x68616559 0x00000000
0xffffd4f8: 0x0068614e 0x00000000 0x68616559 0x00000000
0xffffd508: 0x6279614d 0x00a50065 0x68616559 0x00000000
0xffffd518: 0x6576654e 0x00000072 0x6e6f7257 0x08040067
0xffffd528: 0x6e6f7257 0x08040067 0x00006f4e 0x00000000
0xffffd538: 0x6279614d 0x00a50065 0x6e6f7257 0x08040067
0xffffd548: 0x0068614e 0x00000000 0x68676952 0x08040074
0xffffd558: 0x65727553 0x00000000 0x00006f4e 0x00000000
0xffffd568: 0x68616559 0x00000000 0x0068614e 0x00000000
0xffffd578: 0x0068614e 0x00000000 0x68676952 0x08040074
0xffffd588: 0x00736559 0x00000000 0x68616559 0x00000000
0xffffd598: 0x00006f4e 0x00000000 0x68616559 0x00000000
0xffffd5a8: 0x68616559 0x00000000 0x656b694c 0x0000796c
0xffffd5b8: 0x68676952 0x08040074 0x00006f4e 0x00000000
0xffffd5c8: 0x6576654e 0x00000072 0x6e6f7257 0x08040067
0xffffd5d8: 0x00736559 0x00000000 0x6576654e 0x00000072
0xffffd5e8: 0x0068614e 0x00000000 0x656b694c 0x0000796c
0xffffd5f8: 0x65727553 0x00000000 0x00736559 0x00000000
0xffffd608: 0x65727553 0x00000000 0x65727553 0x00000000
0xffffd618: 0x6576654e 0x00000072 0x656b694c 0x0000796c
0xffffd628: 0x6279614d 0x00a50065 0x6e6f7257 0x08040067
0xffffd638: 0x65727553 0x00000000 0x656b694c 0x0000796c
0xffffd648: 0x6576654e 0x00000072 0x656b694c 0x0000796c
0xffffd658: 0x0068614e 0x00000000 0x00006f4e 0x00000000
0xffffd668: 0x00736559 0x00000000 0x0068614e 0x00000000
0xffffd678: 0x6279614d 0x00a50065 0x6e6f7257 0x08040067
0xffffd688: 0x6279614d 0x00a50065 0x6576654e 0x00000072
0xffffd698: 0x68676952 0x08040074 0x6e6f7257 0x08040067
```

0xffffd6a8:	0x6576654e	0x00000072	0x6279614d	0x00a50065
0xffffd6b8:	0x00006f4e	0x00000000	0x68616559	0x00000000
0xffffd6c8:	0x656b694c	0x0000796c	0x0068614e	0x00000000
0xffffd6d8:	0x0068614e	0x00000000	0x00736559	0x00000000
0xffffd6e8:	0x656b694c	0x0000796c	0x68616559	0x00000000
0xffffd6f8:	0x0068614e	0x00000000	0x68616559	0x00000000
0xffffd708:	0x6279614d	0x00a50065	0x68616559	0x00000000
0xffffd718:	0x6576654e	0x00000072	0x6e6f7257	0x08040067
0xffffd728:	0x6e6f7257	0x08040067	0x00006f4e	0x00000000
0xffffd738:	0x6279614d	0x00a50065	0x6e6f7257	0x08040067
0xffffd748:	0x0068614e	0x00000000	0x68676952	0x08040074
0xffffd758:	0x65727553	0x00000000	0x00006f4e	0x00000000
0xffffd768:	0x68616559	0x00000000	0x0068614e	0x00000000
0xffffd778:	0x0068614e	0x00000000	0x68676952	0x08040074
0xffffd788:	0x00736559	0x00000000	0x68616559	0x00000000
0xffffd798:	0x00006f4e	0x00000000	0x68616559	0x00000000
0xffffd7a8:	0x68616559	0x00000000	0x656b694c	0x0000796c
0xffffd7b8:	0x68676952	0x08040074	0x00006f4e	0x00000000
0xffffd7c8:	0x6576654e	0x00000072	0x6e6f7257	0x08040067
0xffffd7d8:	0x00736559	0x00000000	0x6576654e	0x00000072
0xffffd7e8:	0x0068614e	0x00000000	0x656b694c	0x0000796c
0xffffd7f8:	0x65727553	0x00000000	0x00736559	0x00000000
0xffffd808:	0x65727553	0x00000000	0x65727553	0x00000000
0xffffd818:	0x6576654e	0x00000072	0x656b694c	0x0000796c

Hint – don't forget gdb's trick about reversing byte ordering within each 4-byte chunk.

If the application were to output the value of `magic8ball[3][2]` – what would it be? i.e. what would be returned from the statement `printf("%s", magic8ball[3][2]);`

"Never"