

Problem 14 (2 points)

In the last lab, you were given the following struct definition with an associated typedef.

```
struct noteInfo {
    int frequency;           /* frequency in Hz */
    long duration;          /* duration in mSec */
};
```

Complete the following function, called chipmunk, that doubles the frequency and halves the duration of a struct noteInfo passed using a pointer.

```
void chipmunk(struct noteInfo *note) {
```

*note -> frequency = note -> frequency * 2;*
note -> duration = note -> duration / 2;

```
}
```

Problem 15 (8 points)

In the left column, there are some tricky C expressions. Write their values in the right column. If the values are integers, express them in base 10.

17 % 10	7
17 / 10	1
10 / 5 * 2	4
(17, 15) + 5	20
17 & 14	0
17 14	31
17 && 14	1
17 14	1
~17	-18
!17	0
5 > 4 & 0	0
17 >> 2	4
17 << 2	68
0 && 0 1	1
5 && 0	0
5 0	1

010001 -17
001110 14
000000 0
111111 1

-17 = ~17 + 1
~17 = -18

>>2 17 is divided by 4
<<2 mult by 4
(0 && 0) || 1

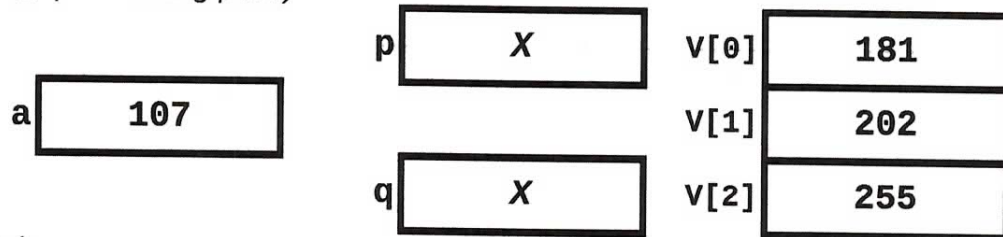
Problem 16 (10 points)

In this question, you are to fill in boxes representing the following C integer or pointer variables to show their values after each of seven sections of C code are executed. You should consider all the sections as being independently executed after the following declaration and initialization statements:

```
int    a = 107 ;
int    V[3] = {181, 202, 255} ;
int    *p = NULL ;
int    *q = NULL ;
```

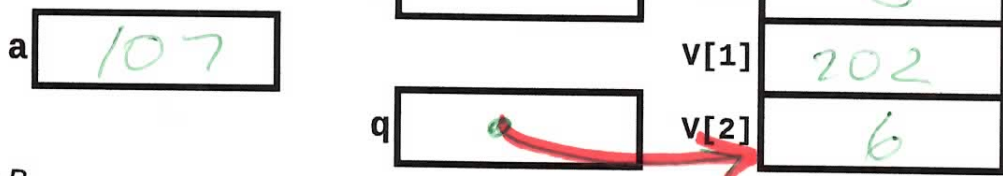
As you might guess, **null** in Java is similar to **NULL** in C. Draw the value **NULL** with a little **X**. Don't ever just leave the pointer variable boxes empty.

Code section @ (the starting point)



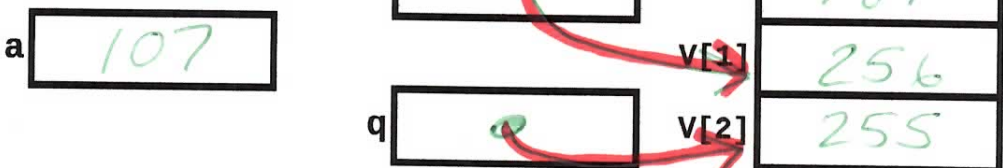
Code section A

```
p = V ;
q = &V[2] ;
*p = 5 ;
*q = 6 ;
```



Code section B

```
p = &V[1] ;
q = &V[2] ;
*p = *q + 1 ;
```



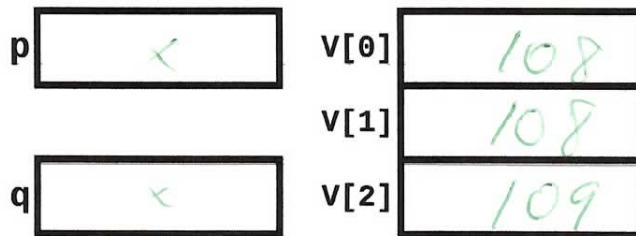
Code section C

```

V[0] = ++a ; a = 108
V[1] = a++ ; a = 109
V[2] = a++ ; a = 110

```

a 110



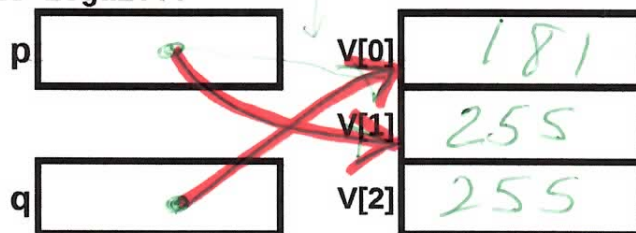
Code section D

```

p = &V[0] ;
q = p++ ;
q[1] = p[1] ;      // This is legal...

```

a



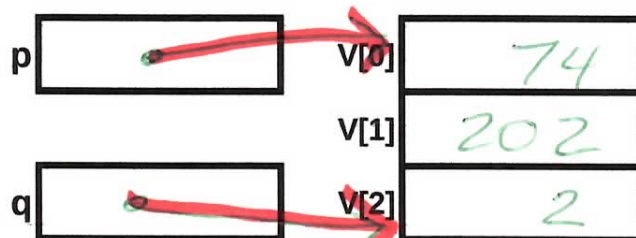
Code section E

```

p = &V[0] ;
q = &V[2] ;
*p = *q - *p ;      255 - 181
*q = q - p ;

```

a



Code section F

```

p = &V[0] ;
q = &V[1] ;
*p++ = 1000 ;      // same as *(p++) = 1000 ;
*++q = 2000 ;

```

a

