Problem 6 (3 points)
In the left column, are eight-bit twos-complement ***fixed point*** numbers with four fractional bits (and consequently four integer bits). Express those numbers in standard decimal point notation in the right column.

01111001	
11010100	

Problem 7 (3 points)
In the left column, are decimal numbers. Express those numbers as eight-bit twos-complement fixed point numbers with four fractional bits (and consequently four integer bits) in the right column.

-5.5	
0.25	

Problem 8 (4 points)
In the left column, there are some tricky C expressions. Write their values in the right column. Express your answers in base 10. (Assume that integers are being represented in 32-bit twos-complement, just like in Java.)

all others are same in Java

not legal in Java

2 * 3 / 5	1
2 / 3 * 5	0
17 | 24	25
17 & 24	16
17 || 24	1
17 && 24	1
! 5	0
~ 5	-6
13 << 2	52
13 >> 2	3

bitwise

non zero values are true

-5 == ~5+1

* 4

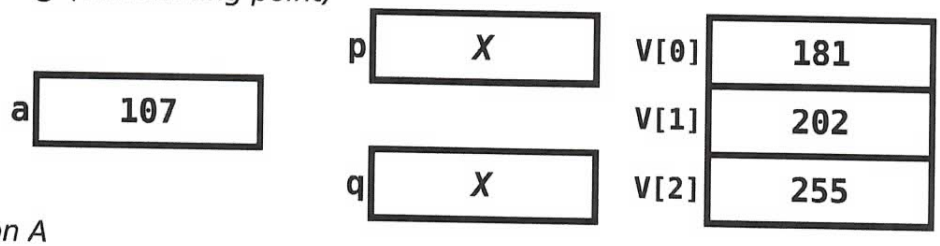
/ 4

Problem 19 (9 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 ;
```

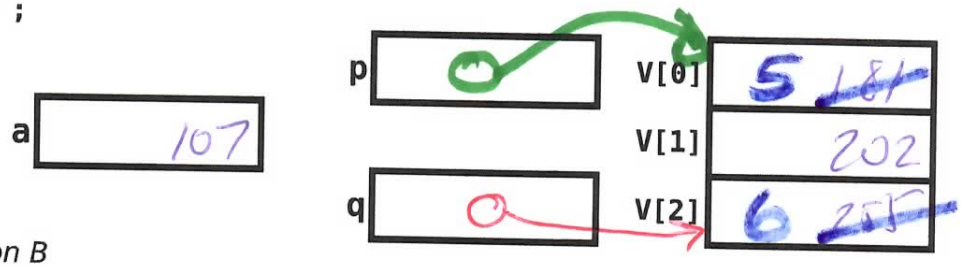
As you might guess, **null** in Java is similar to **NULL** in C. Draw the value **NULL** with a little ***X***. Don't ever just leave the pointer variable boxes empty.

Code section @ (the starting point)



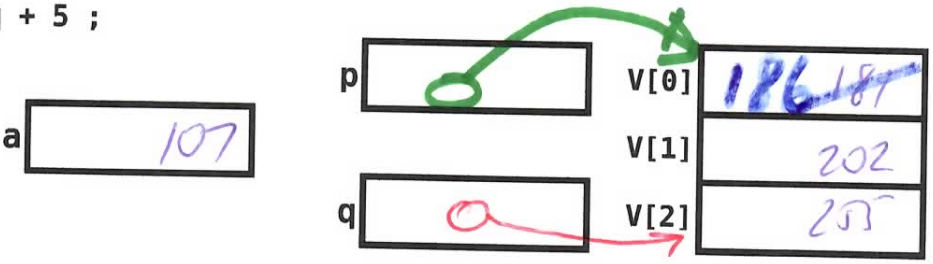
Code section A

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



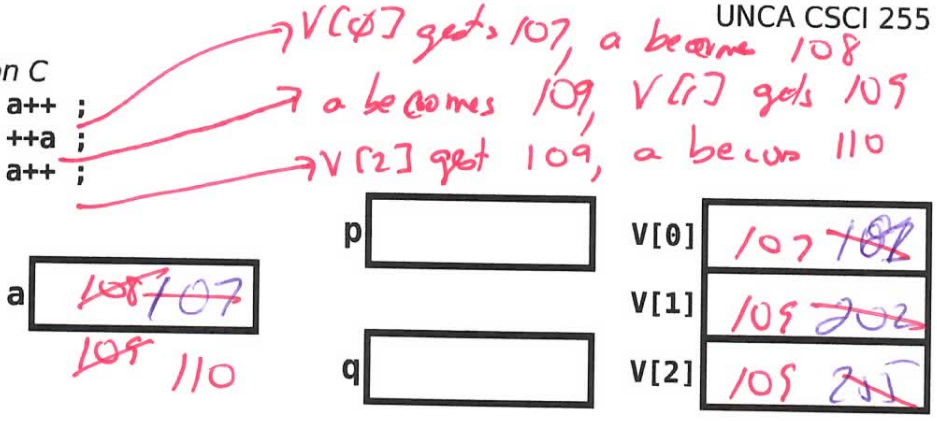
Code section B

```
p = &V[0] ;
q = p + 2 ;
*p = *q + 5 ;
```



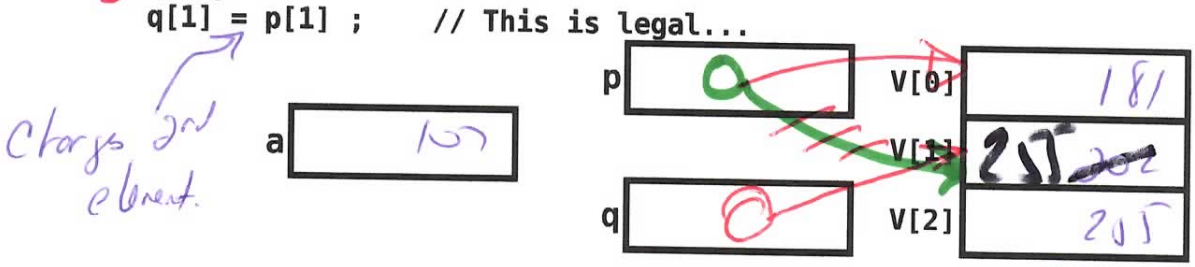
Code section C

```
V[0] = a++ ;
V[1] = ++a ;
V[2] = a++ ;
```



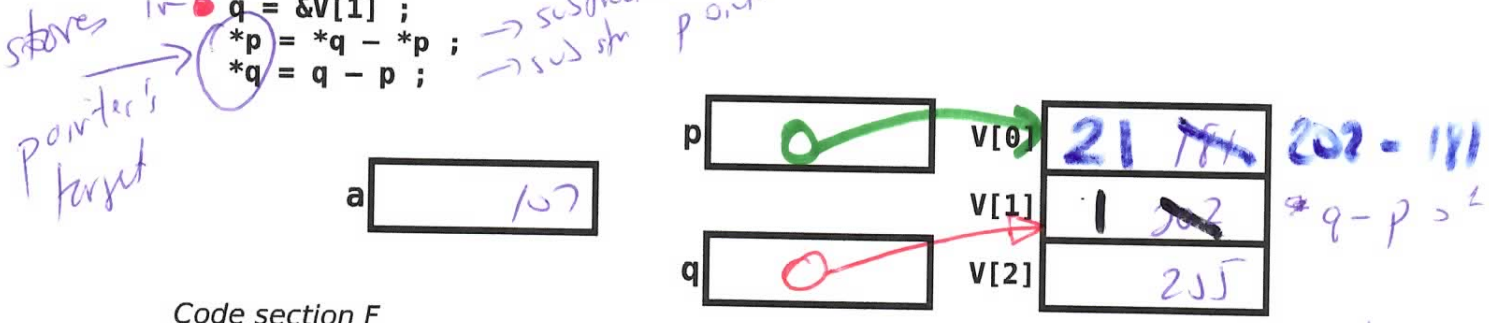
Code section D

```
p = &V[1] ;
q = p-- ;
q[1] = p[1] ;    // This is legal...
```



Code section E

```
p = &V[0] ;
q = &V[1] ;
*p = *q - *p ;
*q = q - p ;
```



Code section F

```
p = &V[0] ;
*p++ = 1000 ;        // same as *(p++) = 1000 ;
a = (*p)++ ;
```