Requirement #1

Solution:

The output is as follows:

```
1: a = 0x7ffc1094cee0, b = 0x55666c7fb2a0, c = (nil)
2: a[0] = 200, a[1] = 101, a[2] = 102, a[3] = 103
3: a[0] = 200, a[1] = 300, a[2] = 301, a[3] = 302
4: a[0] = 200, a[1] = 400, a[2] = 301, a[3] = 302
5: a[0] = 200, a[1] = 128144, a[2] = 256, a[3] = 302
6: a = 0x7ffc1094cee0, b = 0x7ffc1094cee4, c = 0x7ffc1094cee1
```

```
1: a = 0x7ffc1094cee0, b = 0x55666c7fb2a0, c = (nil)
```

Justification:

The memory address of a & b was printed *because* we allocated memory for both pointers. “c” was not allocated in memory, so its value was NIL. We printed the pointers themselves not their contents.

```
2: a[0] = 200, a[1] = 101, a[2] = 102, a[3] = 103
```

Justification:

The for loop assigned each element in the array “a[]” the value “100 + element_Index”, then after that we explicitly assigned “200” to “c[0]”. Note that pointer “c” also points to the array “a[]”.

a[0]=c[0]=200 | a[1]= 100+1 | a[2] =100+2 | a[3]= 100+3

```
3: a[0] = 200, a[1] = 300, a[2] = 301, a[3] = 302
```

Justification:

c[i] ≡ *(c + i) ≡ i[c] → They are all identical and reference the element “i”.

We explicitly assigned these values to the elements. Note that pointer “c” also points to the array “a[]”.

```
4: a[0] = 200, a[1] = 400, a[2] = 301, a[3] = 302
```

Justification:

Pointer “c” was incremented, therefore it points to the second element in the array.

```
5: a[0] = 200, a[1] = 128144, a[2] = 256, a[3] = 302
```

Justification:

We casted pointer “c” to (char*) then when we incremented it it pointed to the next byte (not 4 bytes as expected for int*),then we casted it back to int*. Therefore, when we modified the value of it, a[2] & a[3] were affected because pointer “c” points to 4 bytes (3 bytes in a[2] & 1 byte in a[3])

```
6: a = 0x7ffc1094cee0, b = 0x7ffc1094cee4, c = 0x7ffc1094cee1
```

Justification:

We incremented the pointer “a” by 1, then it moves forward 4 bytes (as it is int*), so pointer “b” points to the location of “a + 4”. Next, we casted pointer “a” to “char*” and incremented it by 1, then it moves forward 1 byte (as it char*), so pointer “c” points to the location of “a +1”.