

1. array[0] = 0x44434241
array[1] = 0x48474645
str0 = qrstuvwx
str1 = yz234560

Contents of Memory starting from 0x7ffd858e3ed0:

0	6	5	4
3	2	z	y
x	w	v	u
t	s	r	q
p	o	n	m
l	k	j	i
48	47	46	45
44	43	42	41

Array[0] (0x7ffd858e3ed0) starts at 41

Array[1] (0x7ffd858e3ed4) starts at 45

str0 (0x7ffd858e3ee0) starts at q

str1 (0x7ffd858e3ef0) starts at y

2. (3^5) is 243, there is 8 left, so $2 \cdot (3^1)$ and $2 \cdot (3^0)$, therefore 100022.
3. 1101//1110//1010//1101//1011//1110//1110//1111//0000 goes 0xdeadbeef0
4. (2^5) is 32, (2^3) is 8, (2^2) is 4, (2^0) is 1, $32+8+4+1 = 45$.
 $32+16+8+4 = 60$
 $16+8+2+1 = 27$
 $8+2+1 = 11$
 $16+1 = 17$
 $32+16+8+4+2+1 = 63$
5. $-32 + 8 + 4 + 1 = -19$
 $-32 + 16 + 8 + 4 = -4$
 $16 + 8 + 2 + 1 = 27$
 $8 + 2 + 1 = 11$
 $16 + 1 = 17$
 $-32 + 16 + 8 + 4 + 2 + 1 = -1$
6. 0x2d
0x3c
0x1b
0x0b

0x11
0x3f

7.

```
arken@Arken: ~/cs224/hw3
1 #include <stdio.h>
2 typedef unsigned char* byte_pointer;
3
4 void show_bytes(byte_pointer start, size_t length)
5 {
6     for (unsigned int i = 0; i < length; ++i)
7     {
8         printf(" %.2x", start[i]);
9     }
10    printf("\n");
11 }
12
13 void show_short(short x)
14 {
15     show_bytes((byte_pointer) &x, sizeof(short));
16 }
17
18 void show_long(long x)
19 {
20     show_bytes((byte_pointer) &x, sizeof(long));
21 }
22
23 void show_double(double x)
24 {
25     show_bytes((byte_pointer) &x, sizeof(double));
26 }
27
28 int main()
29 {
30     show_short(47);
31     show_long(5673);
32     show_double(65.998);
33     return 0;
34 }
```

8.

```
arken@Arken: ~/cs224/hw3
1 #include <stdio.h>
2
3 unsigned replace_byte (unsigned byteWord, int replacePosition, unsigned char replacementByte)
4 {
5     unsigned mask = ~(0xff << replacePosition);
6     unsigned shiftedByte = replacementByte << replacePosition;
7     return ((byteWord & mask) + shiftedByte);
8 }
```

9. A) !!x

B) !!(~x)

C) !(x & 0xff)

D) !!(~x & (0xff << 24))

10.

```
arken@Arken: ~/cs224/hw3
1 #include <stdio.h>
2
3 int main()
4 {
5     int x = 0x55555555;
6     printf("%d\n", !!(x & 0xaaaaaaaa));
7     return 0;
8 }
```

11. A) maxBytes gets implicitly cast to a size_t, so therefore if maxBytes is too small, it will become the hugely positive representation of the negative number resulting from the subtraction.

B) if ((signed)(maxBytes - sizeof(val)) >= 0)

12. A) (x << 4) + (x << 0)

B) (x << 0) - (x << 3)

C) (x << 6) - (x << 2)

D) (x << 4) - (x << 7)

13.

Format A Bits	Value	Format B Bits	Value
1 01111 001	-9/8	1 0111 0010	-9/8
0 10110 011	176	0 1110 0110	176
1 00111 010	-5/1024	1	
0 00000 111	7/131072	0	
1 11100 000	-8192	1	
0 10111 100	384	0	

14. Taylor Whitlock, Evan Smith, Nathan Nelson, Neil Knight; Group 25