1a. Warmup! Convert the following numbers from hex to decimal. Assume two’s complement
0x61
0xc0

b. Let’s go the other way! Convert the decimal numbers to hex!
-20
16

2. Suppose we have a struct list_node_t my_awesome_node and struct linked_list
my_awesome_ll.

struct list_node_t {
    struct list_node_t *next;
    int data;
};

struct linked_list {
    struct list_node_t *head;
};

Assume that the structs are tightly-packed and that we’re in a 32 bit memory address space.

What would sizeof(my_awesome_node) return?
How about sizeof(my_awesome_ll)?

3. Fun with pointers! (Adapted from Sp15, Q1)
Suppose we have the following array

Int arr[] = {0x61c, 0x2010, 0x2, 0xa, 4};
Int *p = arr;

Assuming that integers and pointers are 32 bits, what are the values of the expressions? Write “Error” if an error might occur.
Examine the code:

```c
int a = 5;
void foo() {
    int temp = 4;
    bar();
}

void bar() {
    int hello = 33;
}

int main() {
    int b = 0;
    char* truth = "cs61c is awesome";
    char lie[] = "cs61c sucks";
    char* c = malloc(sizeof(char) * 10);
    foo();

    return 0;
}
```

Where would the following variables live in memory? Code, static, heap, or stack?

<table>
<thead>
<tr>
<th>Variable</th>
<th>Memory Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>truth</td>
<td></td>
</tr>
<tr>
<td>lie</td>
<td></td>
</tr>
<tr>
<td>truth[0]</td>
<td></td>
</tr>
<tr>
<td>lie[0]</td>
<td></td>
</tr>
<tr>
<td>c[0]</td>
<td></td>
</tr>
</tbody>
</table>

b. Sort the following from least to greatest.

b, &temp, &hello, c, &a

5. Katz is a new student in 61c and is just trying to learn the basics, but might have made some mistakes along the way. Help Katz along by answering the following questions: Is ‘whee’ a usable pointer? Is there a memory leak?

```c
char * foo() {
    char *whee = “I love cs61c!”;
    return whee;
}

char * foo_v2() {
    char whee[5];
    whee[0] = ‘w’;
    whee[1] = ‘h’;
    whee[2] = ‘e’;
    whee[3] = ‘e’;
    whee[4] = ‘!’;
    return whee;
}
```
6a. Katz is trying to use the structs defined earlier:

```c
struct list_node_t {
    struct list_node_t *next;
    int data;
};

struct linked_list {
    struct list_node_t *head;
};
```

This is his delete method:

```c
void delete( linked_list *ll) {
    free(ll);
}
```

Is there a problem? If so, what can be done to fix it?

6b. Alas, Katz also needs help with the insert method. Help him!
Make sure to use malloc()!

```c
void insert (int value, int location, linked_list ll) { ....}
```