Read Chapter 1, especially 1.3 and 1.5. 1.4 will serve you well soon.  
Due Tuesday, September 7. (beginning of class)  
Be nice to your TA: Type. Oh, and think before you type. Or edit after you type. Either way.  
Although you *should* try to complete the assignment *on paper*, code snippets are available at:


As described in the syllabus, cite sources.

1. What are the seven OSI layers, in low to high order?
2. Define encapsulation.  
   
   (To save you time and to keep you from writing junk, here is the specious definition in wikipedia. Write something both intelligible and correct; this is neither.)

   In computer networking, encapsulation is a method of designing modular communication protocols in which logically separate functions in the network are abstracted from their underlying structures by inclusion or information hiding within higher level objects.

3. Define multiplexing. Contrast demultiplexing. Why is multiplexing good?
4. What does a packet look like when carrying a piece of a web page across an 802.11 link? That is, what protocol headers are where? (Don’t overdo the details, just get as much as we’ve covered in class.)
5. What is the output of the following listing (on a 386-type processor, 32-bit, LSB), then, why is this tricky:

   ```c
   #include <stdio.h>
   #include <stdlib.h>

   #define F 3
   #define G(x) x
   #define H(x) x+2

   int main()
   {
     printf("%u %u %u %u\n", sizeof(F), G(H(F)), H(G(F)), G(G(F)));
     exit(EXIT_SUCCESS);
   }
   ```

6. What is the output of the following listing, (on a 386-type processor, 32-bit, LSB) then, why is this question tricky?

   ```c
   #include <stdio.h>
   #include <stdlib.h>

   struct m {
     int a, b, c;
     char d, e;
     int f, g, h;
   };
   ```
int main() {
    struct m *s;
    printf("%lu %lu %lu", sizeof(s), sizeof(*s), sizeof(struct m));
    exit(EXIT_FAILURE);
}

7. What is the output of the following listing, (on a 386-type processor, 32-bit, LSB) then, why can the compiler (gcc -Wall -c refresherC4.c) print a warning?

#include <stdio.h>
#include <stdlib.h>

static void m() {
    static int q;
    printf("%d\n", q);
}

static void n() {
    printf("I am not called.\n");
}

int main() {
    int i;
    for(i = 0; i < 5; i++) m();
    exit(EXIT_SUCCESS);
}

8. What is the output of the code in Fig 1 (on its own page) (on a 386-type processor, 32-bit, LSB).

9. Write the body of a find() function to be invoked by inject() as part of Fig 1. The output should be 0 for 3 and 1 for 4 when called at the end. One line should be enough. Points docked for compiler warnings.
```c
#include <stdio.h>
#include <stdlib.h>

struct linked_list {
    struct linked_list *next;
    void *element;
};

typedef int (*operation)(void *elt, void *storage);

int inject (struct linked_list *p, operation op, void *storage) {
    for (; p && op(p->element, storage); p=p->next);
    return ( p!=NULL ? 1 : 0);
}

int sum(void *elt, void *storage) {
    *(int *)storage += *(int *)elt;
    return 1;
}

int dbl(void *elt, void *storage) {
    *(int *)elt *= 2;
    return 1;
}

int find (void *elt, void *storage) {
    /* todo */
}

int main() {
    struct linked_list sto[10];
    int i, storage;
    for (i=0;i<10;i++) {
        sto[i].next = &sto[i+1];
        sto[i].element = malloc(sizeof(int));
        *(int *)sto[i].element = i+1;
    }
    sto[9].next = NULL;
    storage = 0;
    inject (sto, sum, &storage);
    printf("%d\n", storage);
    inject (sto, dbl, &storage);
    inject (sto, sum, &storage);
    printf("%d\n", storage);
    return 3;
    printf("%d\n", inject (sto, find, &storage));
    storage = 4;
    printf("%d\n", inject (sto, find, &storage));
    exit(EXIT_SUCCESS);
}
```

Figure 1: Function pointer example