0. Care and Use:

Choose and answer 4 of the following 7 parts. Be sure to indicate which 4 you want graded.
Presume iLab data type sizes when necessary

a. What is the difference in size between an int pointer and a short pointer?

b. What functionality does less have that more does not?

c. What are the differences between the ">" and "|" commands (not the logical expressions)?

d. Why can't you dereference a void pointer?

e. If the code "(*mystery).thing" works without error, what must 'mystery' point to?

f. What is the point of checking the pointer that malloc() returns?

g. Construct an enumerated type named 'volumes' whose values are pint, quart and gallon, with each equal to the number of pints it contains.

1. Pointer Gymnastics: Pointers like to stay in shape.

Answer ***all*** of the below. Presume iLab datatype sizes when necessary.

a. Explain the functional differences between the two code blocks below:

(What can you do with ptrA that you can not do with ptrB, and vice versa?)

```
int a = 2;
const int * ptrA = &a;
```

```
int b = 2;
int const * ptrB = &b;
```

b. Explain the difference in what is printed by the two code blocks below:

```
int a = 2;
int * p = &a;
printf("%d\n", sizeof(p));
```

```
char b = 'a';
char * q = &b;
printf("%d\n", sizeof(q));
```

c. Note what will be printed by the code blocks below, and explain why:

```
int * ptr = (int*)malloc(sizeof(int)*2);
char * repoint = (char*)ptr;

if(ptr < repoint){ printf("a p<r"\n); }
else if (ptr > repoint){ printf("a p>r\n"); }
else { printf("a p==r\n); }
```

```
ptr = repoint;

if(ptr++ < ++repoint){ printf("b p<r"\n); }
else if (ptr++ > ++repoint){ printf("b p>r\n"); }
else { printf("b p==r\n); }

ptr = repoint;

if(++ptr < ++repoint){ printf("c p<r"\n); }
else if (++ptr > ++repoint){ printf("c p>r\n"); }
else { printf("c p==r\n); }
```

2. Pointing: For fun and profit!

Answer ***all*** of the following.

a. How many bytes long is the string "hello"?

b. Write a pointer named "fn" to a function with the prototype "int compare(void* val0, void* val1)"

c. Based on the code below, determine what is printed after the addition:

```
int * ptr;
printf("addr: %X\n", ptr);    //→ outputs: 0xDB1A98
ptr = ptr+2;
printf("addr: %X\n", ptr);  //→ outputs: _____
```

3. IO & I-Nodes

Answer ***all*** of the following.

a. Why do you have to check the value of a non-blocking IO call?

b. What would happen if you did not increment the address of the buffer you issue a non-blocking write from as you loop on it?

c. A non-blocking IO call will return in two cases. What are they?

d. How can a statically-sized inode with only 30 direct-mapped pointers index a file that would require 100 disk blocks?

e. What is a p-node?

4. TA Time:

The code below is supposed to modify some value pointed to by its parameter and return that pointer. Find, describe and correct all errors.

```
int* someFunc(int * value)
{
        int check = *value;
        int* gentest = (int*)malloc(sizeof(int)*2);
        *gentest = check;

        int newval = ((++check)*2)-12;
        *(gentest+4) = newval;

        check = *(gentest+4);
        free(newval);

        int* retval = &check;

        return retval;
}
```