CS16, UCSB
Pre-lab #7: Worth 50% of Lab 7 score (50 total points)

Print this form, staple loose pages together, and write your answers on it.

Accepted: On paper, in lab Wednesday, August 15.

Name (2 pts): _____

Email (2 pts): _____

Lab section (2 pts) Circle one: 2:00 3:30

If you have the book, read chapter 6. Then answer the following questions.

1. Refer to the following variables for all parts of this problem #1:

```
int a[] = { 5, 10, 15, 20 };
int *p = a;
```

   a. (5 pts) Draw a memory diagram for these variables, but instead of showing a (memory address) value in the diagram for `p`, show an arrow that indicates where `p` is pointing (as the instructor does in lectures).

   b. (5 pts) Show the values of each of the following expressions, or write `ILLEGAL` if the expression is not legal for these variables:

      i. `*p` _____

      ii. `*(p + 2)` _____

      iii. `*p + 2` _____

iv. `*(a + 2)` _____

v. `p - 1` _____

c. (10 pts) Using `p` instead of `a`, and without using any index variable like `i`, write a loop structure that prints all four values of the array `a`. (Hints: loop as long as `p` is less than `a + 4`, dereference `p` inside the loop, and increment `p` at the end of the loop.)

2. If you have the book, read section 6.4 (pgs. 300-302). Then answer the following questions in your own words. Diagrams are encouraged but not required.

   Consider the following code that is intended to swap the values of the two given variables (adapted from page 300):

```
void switch1( int a, int b ) {
 int hold;
 hold = a;
 a = b;
 b = hold;
}
```

   a. (5 pts) Why is `switch1` incorrect? I.e. why does it fail to swap the values of the two variables?

Consider a modification of the above code that does correctly swap the values of two variables (adapted from page 301):

```
void switch2( int* a, int* b ) {
 int hold;
 hold = *a;
 *a = *b;
 *b = hold;
}
```

b. (5 pts) Why does `switch2` correctly solve the problem?

3. If you have the textbook, read section 6.6. Otherwise refer to http://www.cplusplus.com/reference/clibrary/cstring/. Refer to the following variable for both parts of this problem:

```
char word[ 6 ] = "fun";
```

a. (5 pts) Draw a memory diagram for this array, and be sure to show ALL of the array's contents. (Hint: you should show 6 bytes total.)

b. (5 pts) Show the values of each of the following function calls. Show pointers as string suffixes (i.e. the return value of `&(word[ 1 ])` would be `"un"`.

i. `strlen( word )` _____

ii. `strcmp( word, "fun" )` _____

iii. `strchr( word, 'r' )` _____

iv. `strstr( word, "un" )` _____

v. `strcat( word, "ny" )` _____

c. (4 pts) In 3.b, `word` was explicitly declared to have more elements in the character array than there were characters in the underlying representation of `"fun"`. This was important so that we could later call `strcat( word, "ny" )`. Why? (Hint: what would happen if `word` had been instead been declared like: `char[] word = "fun";`?)

Pre-lab End. Adapted from Michael Costanzo by Kyle Dewey.