CS16, 10W, UCSB—H14: (Character Strings, Etter, Section 6.6) Total Points: 50 (printable PDF)
Available online at: http://www.cs.ucsb.edu/~pconrad/cs16/10W/homework/H14
Accepted: **on paper, in Lecture (11am Thursday Mar 4th)**

Name: (3 pts)____________________________________ UMail address (4 pts) _____________@umail.ucsb.edu

Lab Section (3 pts) Circle one: 3pm 4pm 5pm unknown

Name of your pair programming partner(s), if you work together:

This assignment is due **IN Lecture on Thursday 03/04.**
It may ONLY be turned in during Lecture on Thursday.

This assignment requires the use of Ch on CSIL—to complete it you can visit CSIL in person,
access the CSIL command line on "csil.cs.ucsb.edu" via PuTTY on Windows
or use "ssh username@csil.cs.ucsb.edu" on Mac or Linux

Reading: Read Section 6.6 in your Etter textbook about character strings—but as you do, keep these things in mind.

- You should read all of the text on p 308-309 in detail.
- You do not need to read the ENTIRE detailed list of string functions on p. 310—this material belongs in a reference, not a textbook section!
- Please DO read, however, about the following smaller list of important functions:
    - p. 309:
        - strlen(s) returns length of a string
        - strcpy(s,t) string copy, and strncpy(s,t,n) string copy with a limit of n characters
            - Note that strcpy can be a very dangerous function—it is the basis of many security attacks.
            - strncpy is often preferred—we'll talk more about this below.
    - p. 310
        - strcmp(s,t) string compare and strncmp(s,t,n) string compare (with a limit of n characters)
- You may skip the example program on p. 310-311, since it covers strcat, and we aren't going to cover it
- Start your reading again from the words "Character strings are used ..." on p. 311, and look at the function strg_len_2 in some detail.
- You may stop reading when you get to the paragraph at the top of 312 that starts with "Assume that we want to count..."

Now try the following, using Ch:

First, start up Ch—and optionally, simplify your Ch prompt—by doing this:

```
-bash-3.2$ ch
                                    Ch
                Professional edition, version 6.1.0.13751
              (C) Copyright 2001-2009 SoftIntegration, Inc.
                     http://www.softintegration.com
/cs/faculty/pconrad> _prompt="Ch> "
Ch>
```

Next try declaring a character string that contains the letters UCSB, like this:

```
Ch> char s1[] ="UCSB";
Ch>
```

Then, use strlen to return the length of this string, like this:

```
Ch> strlen(s1)
4
Ch>
```

So, we see that strlen can determine the length of a string.

1. The value returned by strlen is 4. However, the true size of the array s1—i.e. the number of elements s1[0], s1[1], etc.—is not 4.

    a. (3 pts) What is the actual size of the array s1?

    b. (3 pts) More importantly—why does s1 have that size?

Please turn over for more problems

This side continues the Ch session started on the other side...

2. Next, try the following command, and note that you get an error message:

```
Ch> char s2[4]="UCSB";
What error message appears here?
Ch>
```

When you type `char s2[4]="UCSB";` into Ch:

   a. (3 pts) What is the error message that you get from Ch?

   b. (3 pts) More importantly—why do you get this error mesage? Explain.

3. Now type the following into Ch:

```
Ch> char s3[16]="UCSB";
Ch>
```

Then, type in strlen(s3)

   a. (2 pts) What is the value that is returned?

   b. (2 pts) Why is this value returned instead of the value 16?

4. Now try typing the following, and note the error message that you get:

```
Ch> s3 = "Cal Poly SLO";

ERROR: invalid lvalue of assignment operation

Ch>
```

The error message refers to an "invalid *lvalue*". As we've discussed in lecture:

- An *lvalue* is a value on the "left hand side of the assignment statement".
- So, the *lvalue* here is `s3`
- Since `s3` is the name of an array, its value is actually a pointer to the `s3[0]` element in that array.
- It is illegal in C to change the name of an array to point to something else—we can change the contents of s3, but not where s3 points.
- So, s3 can never appear on the left hand side of an assignment—it is an invalid *lvalue*.

Instead, to assign the contents of s3 to be something else, we need to use a function call to `strcpy(s,t)`, like this:

```
Ch> strcpy(s3,"Cal Poly SLO");
```

Try that.

   a. (2 pts) What is the value of **`strlen(s3)`** after you use `strcpy(s3,"Cal Poly SLO");` ?

   b. (4 pts) What strcpy command would use use to change the contents of `s3` back to `"UCSB"`?

   c. (2 pts) After typing the command to change s3 back to `"UCSB"`, what would the value of **`strlen(s3)`** be?

5. Start this problem by typing the following into Ch:

```
Ch> char s4[32]="UC San Diego";
Ch>
```

Then type s4 at the prompt, to verify that s4 now contains the string "UC San Diego":

```
Ch> s4
UC San Diego
Ch>
```

Note that Ch does NOT print the value of s4—which is actually a (char *), an address, a pointer to s4[0]—the authors of the Ch software assumed (rightly) that most users would prefer to see the contents of s4.

So, (char *) pointers get special treatment in Ch—when you type one, you see the contents of the string pointed to, not the pointer value. To see s4 as a pointer, we can use a special syntax, shown below—this is called "casting s4 to a void pointer"

```
Ch> (void *) s4
0x85acc28
Ch>
```

With that in mind, continue as follows:

   a. (2 pts) What is the value of **`strlen(s4)`** when you type if at the Ch prompt? Try to predict it before you type it in.

   b. (2 pts) Now, type this: **`s4[2] = '\0';`**

      What is the value shown when you type s4 at the Ch prompt?

   c. (2 pts) As the previous problem showed, the value printed when you type s4 at the Ch prompt has changed. Try typing **`(void *) s4`** at the Ch prompt. Has the value changed?

   d. (2 pts) (continuing from previous problem...) If the value of **`(void *) s4`** changed, why did it? If it did not change, why did it not change?

   e. (2 pts) Now, type **`s4[2] = ' ';`** (that's a space in single quotes) Now what gets printed when you type **`s4`** at the Ch prompt?

   f. (2 pts) And what is the value of **`strlen(s4)`** now?

   g. (4 pts) In your own words, what does this show about the role of the '\0' character in how strings are treated in C?

End of H14