

York University  
Electrical Engineering and Computer Science

EECS2031: Software Tools  
SU2016  
Assignment #7

Chapter 13: Exercises

1. The following function calls supposedly write a single new-line character, but some are incorrect. Identify which calls don't work and explain why.

(a) <code>printf("%c", '\n');</code>	(g) <code>putchar('\n');</code>
(b) <code>printf("%c", "\n");</code>	(h) <code>putchar("\n");</code>
(c) <code>printf("%s", '\n');</code>	(i) <code>puts('\n');</code>
(d) <code>printf("%s", "\n");</code>	(j) <code>puts("\n");</code>
(e) <code>printf('\n');</code>	(k) <code>puts("");</code>
(f) <code>printf("\n");</code>	
2. Suppose that `p` has been declared as follows:  

```
char *p = "abc";
```

Which of the following function calls are legal? Show the output produced by each legal call, and explain why the others are illegal.

(a) <code>putchar(p);</code>
(b) <code>putchar(*p);</code>
(c) <code>puts(p);</code>
(d) <code>puts(*p);</code>
5. (a) Write a function named `capitalize` that capitalizes all letters in its argument. The argument will be a null-terminated string containing arbitrary characters, not just letters. Use array subscripting to access the characters in the string. *Hint:* Use the `toupper` function to convert each character to upper-case.  
(b) Rewrite the `capitalize` function, this time using pointer arithmetic to access the characters in the string.
9. What will be the value of the string `s1` after the following statements have been executed?

```
strcpy(s1, "computer");  
strcpy(s2, "science");  
if (strcmp(s1, s2) < 0)  
    strcat(s1, s2);  
else  
    strcat(s2, s1);  
s1[strlen(s1)-6] = '\0';
```

## Chapter 13: Programming Projects

1. Write a program that finds the “smallest” and “largest” in a series of words. After the user enters the words, the program will determine which words would come first and last if the words were listed in dictionary order. The program must stop accepting input when the user enters a four-letter word. Assume that no word is more than 20 letters long. An interactive session with the program might look like this:

```
Enter word: dog
Enter word: zebra
Enter word: rabbit
Enter word: catfish
Enter word: walrus
Enter word: cat
Enter word: fish
```

```
Smallest word: cat
Largest word: zebra
```

*Hint:* Use two strings named `smallest_word` and `largest_word` to keep track of the “smallest” and “largest” words entered so far. Each time the user enters a new word, use `strcmp` to compare it with `smallest_word`; if the new word is “smaller,” use `strcpy` to save it in `smallest_word`. Do a similar comparison with `largest_word`. Use `strlen` to determine when the user has entered a four-letter word.

16. Modify Programming Project 1 from Chapter 12 so that it includes the following function:

```
void reverse(char *message);
```

The function reverses the string pointed to by `message`. *Hint:* Use two pointers, one initially pointing to the first character of the string and the other initially pointing to the last character. Have the function reverse these characters and then move the pointers toward each other, repeating the process until the pointers meet.

17. Modify Programming Project 2 from Chapter 12 so that it includes the following function:

```
bool is_palindrome(const char *message);
```

The function returns `true` if the string pointed to by `message` is a palindrome.