**Talen Phillips**

**EE 107-01**

**Homework 9**

**Due: 01MAY2013**

1) What will be the value of the string *s1* after the following statements have been executed within a program?

strcpy(s1, “computer”);

strcpy(s2, “science”);

if (strcmp(s1,s2) < 0) strcat(s1,s2);

else strcat(s2,s1);

s1[strlen(s1)-6] = ‘\0’;

**Answer: This should print “computers”.**

2) Write the following function:

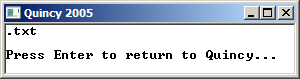
*void get\_extension(const char \*file\_name, char \*extension);*

where *file\_name* points to a string containing a file name. The function should store the extension on the file name in the string pointed to by *extension*. For example , if the file name is “memo.txt”, the function will store “txt” in the string pointed to by *extension*.

I've included the **answer in bold on the following page**.

NOTE: Since the question only asks for the function. The main function does not point to an actual file name.

The program outputs the following:



/\* Talen Phillips

\* EE107-01

\* 29APR2013

\* "Extension array"

\*/

#include <stdio.h>

/\* max extension length +2 (for period and null terminator)\*/

#define EXT\_MAX 6

/\* declare the get\_extension function \*/

void get\_extension(const char \* file\_name, char \*extension);

int main (void){

/\* the "name" array is declared with a specific name for

\* demonstration purposes. \*/

char name[]="memo.txt";

/\* declaring the array that will hold the extension \*/

char ext[EXT\_MAX];

/\* setting both pointers to the first element of the

\* respective arrays \*/

char \*file\_name = &name[0];

char \*extension = &ext[0];

/\* function call \*/

get\_extension(file\_name, extension);

/\* output the ext array for demonstration \*/

puts(ext);

return 0;

}

**void get\_extension(const char \*file\_name, char \*extension){**

**/\* Unless I specifically tell the last loop to stop at**

**\* the end of the array, it will overrun, and begin**

**\* writing into memory directly adjacent to the array.\*/**

**int i;**

**/\* The first loop sets the file\_name pointer to the**

**\* period at the beginning of the extension \*/**

**while(\*file\_name!='.') ++file\_name;**

**/\* The second loop copies the data from the name array**

**\* into the ext array. Note the use of i for safety. \*/**

**for(i=0; (\*file\_name!='\0')&&(i<EXT\_MAX-1); ++i){**

**\*extension=\*file\_name;**

**++extension;**

**++file\_name;**

**}**

**\*extension='\0'; /\* set the null terminator \*/**

**return;**

**}**

3) Write the following function:

*double compute\_average\_word\_length(const char \*sentence);*

which computes and returns the average length of the words in the string pointed to by sentence. Use the function in a program by entering a sentence from the user using getchar() and printing the average word length to the screen.

/\* Talen Phillips

\* EE107-01

\* 29APR2013

\* "mean word length"

\*/

#include <stdio.h>

/\* declare the required function \*/

double compute\_average\_word\_length(const char \*sentence);

int main (void){

int n; /\* loop counter \*/

char sen[101]; /\* array to hold the sentence \*/

char \*sentence; /\* pointer to manipulate the string \*/

sentence = &sen[0]; /\* pointing at object zero of the string \*/

printf("Enter a sentence: \n");

/\* input using getchar (as required) and the pointer. \*/

for (n=0; (n<100)&&(\*(sentence-1)!='\n'); ++n){

\*sentence = getchar();

++sentence;

}

\*sentence = '\0'; /\* adding null terminator \*/

sentence = &sen[0]; /\* resetting pointer \*/

/\* calling function, and printing result \*/

printf("Mean word length is %g",compute\_average\_word\_length(sentence));

return 0;

}

**double compute\_average\_word\_length(const char \*sentence){**

**double words,letters;**

**for (words=0,letters=0; \*sentence!='\0'; ++sentence){**

**/\* If the current character isn't a letter and the previous one was,**

**\* increment words. \*/**

**if (((\*sentence<'a' || \*sentence>'z')&&(\*sentence<'A' || \*sentence>'Z')) && ((\*(sentence-1)>='a' && \*(sentence-1)<='z')||(\*(sentence-1)>='A' && \*(sentence-1)<='Z')))**

**++words;**

**/\* Otherwise, if the current character is a letter, increment letters \*/**

**else if ((\*sentence>='a' && \*sentence<='z') || (\*sentence>='A' && \*sentence<='Z'))**

**++letters;**

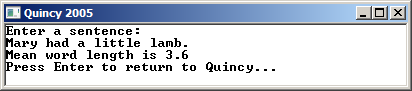
**}**

**/\*return the average\*/**

**return (letters/words);**

**}**

Output on following page.



4) Suppose the f and p are declared as follows:

struct {

union {

char a,b;

int c;

} d;

int e[5];

} f, \*p = &f;

Which of the following statements are legal?

(a) p->b = ‘ ‘;

(b) p->e[3] = 10;

(c) (\*p).d.a = ‘\*’;

(d) p->d->c = 20;

**Statements (a) and (b) are legal.**