Assignment 8

**(Displaying Strings in Uppercase and Lowercase**) Write a program that inputs a line of text

into char array s[100]. Output the line in uppercase letters and in lowercase letters.

Enter a line of text:

A line with UPPER- and lowercase LeTters

The line in uppercase is:

A LINE WITH UPPER- AND LOWERCASE LETTERS

The line in lowercase is:

a line with upper- and lowercase letters

(**Converting Strings to Integers for Calculations**) Write a program that inputs four strings

that represent integers, converts the strings to integers, sums the values and prints the total of the

four values.

Enter an integer string: 43

Enter an integer string: 77

Enter an integer string: 120

Enter an integer string: 9999

The total of the values is 10239

(**Random Sentences**) Write a program that uses random number generation to create sentences.

The program should use four arrays of pointers to char called article, noun, verb and preposition.

The program should create a sentence by selecting a word at random from each array in

the following order: article, noun, verb, preposition, article and noun. As each word is picked,

it should be concatenated to the previous words in an array large enough to hold the entire sentence.

The words should be separated by spaces. When the final sentence is output, it should start with a

capital letter and end with a period. The program should generate 20 such sentences. The arrays should be filled as follows: The article array should contain the articles "the", "a", "one", "some"

and "any"; the noun array should contain the nouns "boy", "girl", "dog", "town" and "car"; the

verb array should contain the verbs "drove", "jumped", "ran", "walked" and "skipped"; the preposition

array should contain the prepositions "to", "from", "over", "under" and "on".

After the preceding program is written and working, modify it to produce a short story consisting of several of these sentences. (How about the possibility of a random term paper writer?)

A dog skipped to any car.

Some town ran on the boy.

A dog jumped from the dog.

One girl jumped on one town.

One dog jumped from some boy.

One girl jumped under any dog.

One car drove on some girl.

One town walked on a girl.

Some town ran on one dog.

One car walked from any town.

A boy drove over some girl.

The dog skipped under a boy.

The car drove to a girl.

Some town skipped under any car.

A boy jumped from a town.

Any car jumped under one town.

Some dog skipped from some boy.

Any town skipped to one girl.

Some girl jumped to any dog.

(**Displaying a Sentence with Its Words Reversed**) Write a program that inputs a line of text,

tokenizes the line with function strtok and outputs the tokens in reverse order.

Enter a line of text:

testing 1 2 3

The tokens in reverse order are:

3 2 1 testing

(**Searching for Substrings**) Write a program that inputs a line of text and a search string

from the keyboard. Using function strstr, locate the first occurrence of the search string in the line

of text, and assign the location to variable searchPtr of type char \*. If the search string is found,

print the remainder of the line of text beginning with the search string. Then, use strstr again to

locate the next occurrence of the search string in the line of text. If a second occurrence is found,

print the remainder of the line of text beginning with the second occurrence. [Hint: The second call

to strstr should contain searchPtr + 1 as its first argument.]

Enter a line of text:

To be or not to be; that is the question.

Enter a search string: be

The remainder of the line beginning with

the first occurrence of "be":

be or not to be; that is the question.

The remainder of the line beginning with

the second occurrence of "be":

be; that is the question.

(**Strings Ending with "ed")** Write a program that reads a series of strings and prints only

those strings that end with the letters “ed.”

Enter a string: WALKED

Enter a string: SKIPPED

Enter a string: JUMPED

Enter a string: FLEW

Enter a string: DROVE

The strings ending with "ED" are:

WALKED

SKIPPED

JUMPED

(**Counting the Occurrences of a Character**) Write a program that inputs several lines of text

and a search character and uses function strchr to determine the total occurrences of the character

in the lines of text.

Enter three lines of text:

This program inputs three lines of text

and counts the number of occurrences of

the specified search character in the text

Enter a search character: e

The total occurrences of 'e' in the text is 15

(**Alphabetizing a List of Strings**) Use the string-comparison functions discussed in

Section 8.6 and the techniques for sorting arrays developed in Chapter 6 to write a program that

alphabetizes a list of strings. Use the names of 10 or 15 towns in your area as data for your program.

Enter a string: Westborough

Enter a string: Wellesley

Enter a string: Natick

Enter a string: Waltham

Enter a string: Framingham

Enter a string: Marlborough

Enter a string: Boston

Enter a string: Ashland

Enter a string: Hopkington

Enter a string: Shrewsbury

The strings in sorted order are:

Ashland

Boston

Framingham

Hopkington

Marlborough

Natick

Shrewsbury

Waltham

Wellesley

Westborough

**(Printing Dates in Various Formats)** Dates are commonly printed in several different formats

in business correspondence. Two of the more common formats are

07/21/2003 and July 21, 2003

Write a program that reads a date in the first format and prints it in the second format.

Enter a date in the form mm/dd/yyyy: 06/18/2003

The date is: June 18, 2003

**(Writing the Word Equivalent of a Check Amount)** Continuing the discussion of the previous

example, we reiterate the importance of designing check-writing systems to prevent alteration

of check amounts. One common security method requires that the check amount be both written

in numbers and “spelled out” in words. Even if someone is able to alter the numerical amount of

the check, it is extremely difficult to change the amount in words.

Many computerized check-writing systems do not print the amount of the check in words.

Perhaps the main reason for this omission is the fact that most high-level languages used in commercial

applications do not contain adequate string-manipulation features. Another reason is that

the logic for writing word equivalents of check amounts is somewhat involved.

Write a program that inputs a numeric check amount and writes the word equivalent of the

amount. For example, the amount 112.43 should be written as

ONE HUNDRED TWELVE and 43/100

Enter the check amount ( 0.00 to 99.99 ): 72.63

The check amount in words is:

SEVENTY-TWO and 63/100

Enter the check amount ( 0.00 to 99.99 ): 13.22

The check amount in words is:

THIRTEEN and 22/100

Enter the check amount ( 0.00 to 99.99 ): 5.75

The check amount in words is:

FIVE and 75/100

**Write a program that reads several lines of text** and prints a table indicating the number

of occurrences of each different word in the text. The first version of your program

should include the words in the table in the same order in which they appear in the text.

A more interesting (and useful) printout should then be attempted in which the words

are sorted alphabetically. For example, the lines

To be, or not to be: that is the question:

Whether 'tis nobler in the mind to suffer

contain the words “to” three times, the word “be” two times, the word “or” once, and

so on.

Enter three lines of text:

This program counts the number

of occurrences of each word in

the input text.

"This" appeared 1 time

"program" appeared 1 time

"counts" appeared 1 time

"the" appeared 2 times

"number" appeared 1 time

"of" appeared 2 times

"occurrences" appeared 1 time

"each" appeared 1 time

"word" appeared 1 time

"in" appeared 1 time

"input" appeared 1 time

"text" appeared 1 time

**Write a program that reads several lines of text** and prints a table indicating the number

of occurrences of each letter of the alphabet in the text. For example, the phrase

To be, or not to be: that is the question:

contains one “a,” two “b’s,” no “c’s,” and so on.

Enter three lines of text:

This program counts the occurrences of each

letter of the alphabet in the input text. Then,

it prints a summary of the occurrences.

Total letter counts:

a: 6

b: 1

c: 8

d: 0

e: 14

f: 3

g: 1

h: 8

i: 5

j: 0

k: 0

l: 2

m: 3

n: 7

o: 7

p: 4

q: 0

r: 9

s: 6

t: 15

u: 5

v: 0

w: 0

x: 1

y: 1

z: 0