**LESSON SET 10**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Characters and Strings**

**OBJECTIVES FOR STUDENT**

**Lesson 10A:**

1. To demonstrate the unique characteristics of character data

2. To view strings as an array of characters

3. To show how to input and output strings

**Lesson 10B:**

1. To work with string functions

**ASSUMPTIONS**

It is essential that students have a solid understanding of character arrays before

attempting these labs. Completion of Lesson Set 7 is a sufficient pre-requisite for most exercises in this lab. Lesson Set 9 is certainly more than a sufficient prerequisite for this lesson. However, if you decide to skip Lesson Set 9, then gives a brief

explanation of the functions that use pointers about Lab 10.1. The last section of the Pre-lab

Reading Assignment (Pointers and Strings) may also require a few comments from the instructor if Lesson Set 9 was skipped.

**Lesson 10A:**

1. Students have a good understanding of the pre-lab reading material
2. Students understand basic character functions as given in the pre-lab reading material

3. Students understand the use of get() and getline() functions.

**Lesson 10B**:

1. Students understand basic string functions.

**PRE-LAB WRITING ASSIGNMENT SOLUTIONS**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. B

2. boolean

3. 0 (boolean false)

4. #

5. null character \0

6. 24

7. 22

8. In the Gardenof Eden (Note that there is no space between “Garden” and

“of”)

9. <cctype>

10. double-quotation marks, single-quotation marks

**LAB ASSIGNMENTS**

**Lesson 10A:**

Lab 10.1: Character testing and string validation

Lab 10.2: Case conversion

Lab 10.3: Using getline() & get()

**Lesson 10B:**

Lab 10.4: String functions—strcat

Lab 10.5: Student generated code assignments

**LESSON 10A**

**LAB 10.1: Character Testing and String Validation**

NOTE: This program does use pointers minimally in a few of the functions.

Lab 10.1 requires the student to use program american\_equities.cpp from

the Lab 10 folder. This program determines whether or not an entered password

is valid. Valid passwords consist of 8 characters, 5 of which must be letters and

the other 3 digits. The letters and digits can be arranged in any order. In Exercise

1 students are asked to run the program several times using both valid and invalid

passwords. In the next exercise students are asked to alter the program so that

a valid password consists of 10 characters, 6 of which must be digits and the other

4 letters. Exercise 3 requires students to modify their program so that only lowercase letters are allowed for valid passwords.

A solution program for Exercise 2 named american\_equitiesKEY.cpp can be found in the instructor’s folder for Lesson Set 10.

A solution program for Exercise 3 named american\_equitiesKEY2.cpp can be found in the instructor’s folder for Lesson Set 10.

**LAB 10.2: Case Conversion**

For this lab students need to bring in case\_convert.cpp from the Lab 10 folder

(this is Sample Program 10.2 from the Pre-lab Reading Assignment.) This program

shows how the toupper and tolower functions can be applied in a C++

program. The execution of the program is controlled by a do-while loop. Students

are asked to remove this loop and replace it with three lines of code (which are

given to them) to see the effect on the execution of the program. They should

see that the alteration affects execution when invalid data is entered. Students are

also required to replace toupper in the program with tolower without altering

the task being performed. Remind the students that when running this program,

the user input must be integers (nearest dollar spent on food each week).

A solution program can be found in case\_convertKEY.cpp in the instructor’s folder

for Lesson Set 10.

**LAB 10.3: Using getline()**

This lab is a student generated code exercise. Students are asked to write a short

program that defines a character array containing 10 characters. The user should

be prompted to enter their last name using no more than 9 characters. The program

must then read the name into the array and echo print the name back to

the screen. In the second exercise students are asked to run their program and

enter the name **Newmanouskous** at the prompt. The point of this exercise is for

students to see what can happen if they try to put a string into an array that is

too large to hold it. The results may vary depending on the system they are

using. In the third exercise students are asked to re-write the program using the

getline() function. Students will then run their new program and enter

**Newmanouskous** again. They should find that the output is truncated (i.e.,

**Newmanous** is printed to the screen).

A solution program can be found in readdataKEY.cpp in the instructor’s folder for Lesson Set 10.

The fourth exercise asks the students to bring in grades.cpp and the data file grades.txt from the Lab 10 folder. They are asked to fill in the code to output that data to the screen. The data has both character and numeric data types that need to be read. This is a common and not difficult skill that students must have.

A solution program can be found in gradesKEY.cpp in the instructor’s folder for Lesson Set 10.

**LESSON 10B**

**LAB 10.4: String Functions—strcat**

This lab begins with a given portion of code involving two strings. Students are

asked to write a complete program using the given code that outputs the concatenation

of the two strings. Most of the code necessary is given—they only

need to add a few lines. Exercise 2 asks the students to decrease the size of the

array for the first array so that it is no longer large enough to hold the concatenation.

After running their modified program, students should record what happens

(this may vary from system to system).

A solution program can be found in stringKEY.cpp in the instructor’s folder for Lesson Set 10.

**LAB 10.5: Student Generated Code Assignment**

Notice that this lab consists of exercises and not options. Time permitting, the student should do at least Exercise 1 and Exercise 2. Exercise 1 is the classic palindrome program. A palindrome is a string of characters that reads the same forward as backward. Students are asked to write a program that prompts the user to input a string of size 50 characters or less (so they should define an array with 51 elements). The program should determine whether or not the entered string is a palindrome. Students will need to determine the length of the input string before they start comparing elements of the array.

A solution program can be found in palidromKEY.cpp in the instructor’s folder for Lesson Set 10.

Exercise 2 asks the students to compare two names and print them in alphabetical order. It requires knowledge of the strcmp() function.

A solution program can be found in namecompareKEY.cpp in the instructor’s folder for Lesson Set 10.

Exercise 3 is optional and asks the students to determine how many consonants are in an entered string of 50 characters or less. One way the students can do this is to count all of the letters, count the vowels, and then subtract.

A solution program can be found in countConsKEY.cpp in the instructor’s folder for Lesson 10.