Lab 10

LAB 10.1 Character Testing and String Validation

The American Equities investment company offers a wide range of investment opportunities ranging from mutual funds to bonds. Investors can check the value of their portfolio from the American Equities' web page. Information about personal portfolios is protected via encryption and can only be accessed using a password. The American Equities company requires that a password 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. For example,

rt56AA7q
123actyN
1Lo0Dwa9
myNUM741

are all valid passwords. However, the following are all invalid:

```
the476NEw // It contains more than 8 characters (also more than 5
             letters)
be68moon  // It contains less than 3 digits.
$retrn99  // It contains only 2 digits and has an invalid character ('$')
```

American Equities needs a program for their web page that determines whether or not an entered password is *valid*. The program `american_equities.cpp` from the Lab 10 folder performs this task. The code is the following:

```
// This program tests a password for the American Equities
// web page to see if the format is correct

// Place Your Name Here

#include <iostream>
#include <cctype>
#include <cstring>
using namespace std;

//function prototypes
bool testPassWord(char[]);
int countLetters(char*);
int countDigits(char*);

int main()
{
   char passWord[20];

   cout << "Enter a password consisting of exactly 5 "
    << "letters and 3 digits:" << endl;

   cin.getline(passWord, 20);

   if (testPassWord(passWord))
          cout << "Please wait - your password is being verified" << endl;

   else
   {
          cout << "Invalid password. Please enter a password "
               << "with exactly 5 letters and 3 digits" << endl;
          cout << "For example, my37RuN9 is valid" << endl;
   }
```

```
   // Fill in the code that will call countLetters and
   // countDigits and will print to the screen both the number of
   // letters and digits contained in the password.

   return 0;
}

//*************************************************************
// testPassWord
//
//  task:        determines if the word in the
//                character array passed to it, contains
//                exactly 5 letters and 3 digits.
//  data in:       a word contained in a character array
//  data returned: true if the word contains 5 letters & 3
//                digits, false otherwise
//
//*************************************************************
bool testPassWord(char custPass[])
{
   int numLetters, numDigits, length;

   length = ...         // Use strlen() to determine length of string
   numLetters = countLetters(custPass);
   numDigits = countDigits(custPass);

   if (numLetters == 5 && numDigits == 3 && length == 8)
          return true;
   else
          return false;
}

//*****************************************************************
// countLetters
//
//  task:        This function counts the number of letters
//                (both capital and lower case) in the string
//  data in:       pointer that points to an array of characters
//  data returned: number of letters in the array of characters
//
//*****************************************************************

int countLetters(char *strPtr)
{
   int occurs = 0;

   while (*strPtr != '\0')    // loop is executed as long as
   // the pointer strPtr does not point
   // to the null character which
   // marks the end of the string
   {
      if ( ... )          // Use isalpha() to determine if the character is a letter
          occurs++;

      strPtr++;
   }

   return occurs;
}

//*****************************************************************
// countDigits
//
//  task:        This function counts the number of digits
//                in the string
//  data in:       pointer that points to an array of characters
//  data returned: number of digits in the array of characters
```

```
//
//*****************************************************************

int countDigits(char *strPtr)
{
   int occurs = 0;            // this keeps track how many times a digit occurs

   while (*strPtr != '\0')
   {
      if ( ... )       // Use isdigit() to determine if the character is a digit
          occurs++;

      strPtr++;
   }
   return occurs;
}
```

Exercise 1

Fill in the code at parts commented in bold and then run the program several times, testing with both valid and invalid passwords.

Read through the program and make sure you understand the logic of the code.

Exercise 2

Alter the program so that a valid password now consists of 10 characters, 6 of which must be digits and the other 4 letters.

Exercise 3

Adjust your program from Exercise 2 so that only lower case letters are allowed for valid passwords.

LAB 10.2 Case Conversion

Bring in `case_convert.cpp` from the Lab 10 folder. The code is the following:

```
// This program shows how the toupper and tolower functions can be
// applied in a C++ program

// PLACE YOUR NAME HERE

#include <iostream>
#include <cctype>
#include <iomanip>
using namespace std;

int main()
{
   int week, total, dollars;
   float average;
   char choice;

   cout << showpoint << fixed << setprecision(2);

   do
   {
          total = 0;

          for (week = 1; week <= 4; week++)
          {
                 cout << "How much (to the nearest dollar) did you"
                      << " spend on food during week " << week
                      << " ?:" << endl;
                 cin >> dollars;
```

```
                total = total + dollars;
        }

        average = total / 4.0;

        cout << "Your weekly food bill over the chosen month is $"
             << average << endl << endl;

        do
        {
                cout << "Would you like to find the average for "
                     << "another month?";
                cout << endl << "Enter Y or N" << endl;
                cin >> choice;

        } while (toupper(choice) != 'Y' && toupper(choice) != 'N');

   } while (toupper(choice) == 'Y');

   return 0;
}
```

Exercise 1
Run the program several times with various inputs.

Exercise 2
Notice the following do-while loop which appears near the end of the program:

```
do
{
   cout << "Would you like to find the average for another month?";
   cout << endl << "Enter Y or N" << endl;
   cin >> choice;
} while(toupper(choice) != 'Y' && toupper(choice) != 'N');
```

How would the execution of the program be different if we removed this loop? Try removing the loop but leave the following lines in the program:

```
cout << "Would you like to find the average for another month?";
cout << endl << "Enter Y or N" << endl;
cin >> choice;
```

Record what happens when you run the new version of the program.

Exercise 3
Alter program `case_convert.cpp` so that it performs the same task but uses `tolower` rather than `toupper`.

LAB 10.3 Using `getline()` & `get()`

Exercise 1
Write a short program called `readata.cpp` that defines a character array `last` which contains 10 characters. Prompt the user to enter their last name using no more than 9 characters. The program should then read the name into `last` and then output the name back to the screen with an appropriate message. Do not use the `getline()` or `get()` functions!

Exercise 2
Once the program in Exercise 1 is complete, run the program and enter the name **Newmanouskous** at the prompt. What, if anything, happens? (Note that the results could vary depending on your system).

Exercise 3

Re-write the program above using the `getline()` function (and only allowing 9 characters to be input). As before, use the character array `last` consisting of 10 elements. Run your new program and enter **Newmanouskous** at the prompt. What is the output?

Exercise 4

Bring in program `grades.cpp` and `grades.txt` from the Lab 10 folder. Fill in the code in the sections in bold so that the data is properly read from `grades.txt` and the desired output to the screen is as follows:

OUTPUT TO SCREEN

```
Adara Starr      has a(n) 94 average
David Starr      has a(n) 91 average
Sophia Starr     has a(n) 94 average
Maria Starr      has a(n) 91 average
Danielle DeFino  has a(n) 94 average
Dominic DeFino   has a(n) 98 average
McKenna DeFino   has a(n) 92 average
Taylor McIntire  has a(n) 99 average
Torrie McIntire  has a(n) 91 average
Emily Garrett    has a(n) 97 average
Lauren Garrett   has a(n) 92 average
Marlene Starr    has a(n) 83 average
Donald DeFino    has a(n) 73 average
```

DATA FILE

```
Adara Starr          94
David Starr          91
Sophia Starr         94
Maria Starr          91
Danielle DeFino      94
Dominic DeFino       98
McKenna DeFino       92
Taylor McIntire      99
Torrie McIntire      91
Emily Garrett        97
Lauren Garrett       92
Marlene Starr        83
Donald DeFino        73
```

The code of `grades.cpp` is as follows:

```
#include <fstream>
#include <iostream>
using namespace std;

// PLACE YOUR NAME HERE

const int MAXNAME = 20;

int main()
{
   ifstream inData;
   inData.open("grades.txt");

   char name[MAXNAME + 1];    // holds student name
   float average;                    // holds student average

   inData.get(name, MAXNAME + 1);

   while (inData)
   {
      inData >> average;

      // Fill in the code to print out name and
      // student average

      // Fill in the code to complete the while
      // loop so that the rest of the student
      // names and average are read in properly
   }

   return 0;
}
```

LAB 10.4 String Functions – `strcat`

Consider the following code:

```
char string1[25] ="Total Eclipse ";
char string2[11] = "of the Sun";
cout << string1 << endl;
cout << string2 << endl;
strcat(string1,string2);
cout << string1 << endl;
```

Exercise 1

Write a complete program including the above code that outputs the concatenation of `string1` and `string2`. Run the program and record the result.

Exercise 2

Alter the program in Exercise 1 so that `string1` contains 20 characters rather than 25. Run the program. What happens?

LAB 10.5 Student Generated Code Assignments

Exercise 1

A **palindrome** is a string of characters that reads the same forwards as backwards. For example, the following are both palindromes:

1457887541 **madam**

Write a program that prompts the user to input a string of a size 50 characters or less. Your program should then determine whether or not the entered string is a palindrome. A message should be displayed to the user informing them whether or not their string is a palindrome.

Exercise 2

The `strcmp(string1,string2)` function compares `string1` to `string2`. It is a value returning function that returns a negative integer if `string1 < string2`, 0 if `string1 == string2`, and a positive integer if `string1 > string2`. Write a program that reads two names (last name first followed by a comma followed by the first name) and then prints them in alphabetical order. The two names should be stored in separate character arrays holding a maximum of 25 characters each. Use the `strcmp()` function to make the comparison of the two names. Remember that `'a' < 'b'`, `'b' < 'c'`, etc. Be sure to include the proper header file to use `strcmp()`.

Sample Run 1 (inputs are underlined):

```
Please input the first name
Brown, George
Please input the second name
Adams, Sally
The names are as follows:
Adams, Sally
Brown, George
```

Sample Run 2 (inputs are underlined):

```
Please input the first name
Brown, George
```

```
Please input the second name
Brown, George
The names are as follows:
Brown, George
Brown, George
The names are the same
```

Exercise 3

Write a program that determines how many consonants are in an entered string of 50 characters or less. Output the entered string and the number of consonants in the string.