Case Study: String Manipulation

You have been hired as a contract programmer by the Smithfield Natural Gas company. Your first job is to write a program that prints a form letter to customers with an overdue account. The letter should have the form shown in Figure 1. When the letter is printed by your program, however, the fields shown in brackets will be replaced by actual values.

Figure 1 Format of the form letter

```
Dear <Salutation> <Last-Name>:

Our records show that your account has a balance of $<Balance> and
a past-due amount of $<Past-Due>. Your last payment was on <Date>.
Since we haven't heard from you in some time, would you please take
a moment to send us a check for the past-due amount? We value your
business and look forward to serving you in the future.

Sincerely,
The Management

P.S. If you've already sent your payment, ignore this reminder.
```

Inside the letter, the fields listed in Table 1 are shown in brackets.

Table 1 Form letter fields

Field	Description
`Salutation`	Salutation (either Mr. or Ms.)
`Last-Name`	The customer's last name
`Balance`	The customer's total account balance
`Past-Due`	The amount the account is past due
`Date`	The date the customer last made a payment

Before the letter is printed, your program should ask the user to enter values for the fields listed in Table 1. The values should then be inserted into the form letter as it is being printed. The program should perform word-wrap, which means the sentences should be adjusted so no word is split between two lines. Additionally, the letter should have ten-character left and right margins.

Variables

Table 2 lists the major variables needed.

Table 2 Variables

Variable	Description
`part1...part8`	Eight constant global character arrays that hold portions of the form letter
`salutation`	A character array to hold the salutation as a null-terminated C-string
`lastName`	A character array to hold the customer's last name as a null-terminated C-string
`lastPayment`	A character array to hold the date of the last payment as a null-terminated C-string
`balance`	A character array to hold the text representation of the account balance as a null-terminated C-string
`pastDue`	A character array to hold the text representation of the past due amount as a null-terminated C-string
`again`	A character to hold the user's Y or N response when asked if they wish to print another letter
`position`	An integer that holds the printing position. Each time a character is printed, this variable is incremented. It is used to determine when the end of the line is near

Global Variables

The program uses eight constant global character arrays to hold the sections of the form letter that are always the same. The arrays, named `part1` through `part8`, are defined and initialized as follows:

```
const char part1[] = "Dear ";
const char part2[] = "Our records show that your account has a"
                     " balance of $";
const char part3[] = " and a past due amount of $";
const char part4[] = "Your last payment was on ";
const char part5[] = "Since we haven't heard from you in some"
                     " time, would you please take a moment to send"
                     " us a check for the past due amount? We value"
                     " your business and look forward to serving you"
                     " in the future.\n\n";
const char part6[] = "Sincerely,\n";
const char part7[] = "The Management\n\n";
const char part8[] = "P.S. If you've already sent your payment, ignore"
                     " this reminder.";
```

NOTE: Notice some of the arrays are initialized with what appears to be more than one string. For instance, look at the initialization of `part2`:

```
const char part2[] = "Our records show that your account has a"
                     " balance of $";
```

The two strings are actually concatenated into one string. This allows the programmer to easily span multiple lines when initializing character arrays with long strings. The same technique is used with `part5` and `part8`.

Modules

The program will consist of the functions listed in Table 3.

Table 3 Functions

Function	Description
`main`	The program's `main` function. Calls the `getInfo` and `printLetter` functions.
`getInfo`	Calls the `getSal` function to get the salutation. Then asks the user to enter the customer's last name, account balance, past-due amount, and date of last payment.
`getSal`	Prints a menu allowing the user to select either Mr. or Ms. as the salutation.
`printLetter`	Controls the printing of the form letter once the fields have been input by the user. Calls the `printLine` function.
`printLine`	Prints a line of text starting at the current printing position. This function performs word-wrap when near the end of the line. It keeps the `position` variable updated as well.

Function `main`

Function `main` contains the array definitions for the salutation, last name, date of last payment, account balance, and past-due amount. A `do-while` loop calls the `getInfo` and `printLetter` functions. The loop repeats as long as the user wishes to print form letters. Here is the pseudocode:

```
Do
Call getInfo to get the salutation, last name, balance, past-due
amount, and date of last payment from the user.
   Call printLetter to print the form letter.
   Ask the user if another letter is to be printed.
While the user wants to print another letter.
```

Here is the function's actual C++ code:

```
int main()
{
   // Constants for array sizes
   const int SAL_SIZE = 4;           // salutation size
   const int LNAME_SIZE = 16;        // lastName size
   const int LPAYMENT_SIZE = 16;     // lastPayment size
   const int BAL_SIZE = 9;           // balance size
   const int PDUE_SIZE = 9;          // pastDue size

   char salutation[SAL_SIZE];        // To hold the salutation
   char lastName[LNAME_SIZE];        // Customer's last name
   char lastPayment[LPAYMENT_SIZE];  // Date of last payment
   char balance[BAL_SIZE];           // Account balance
   char pastDue[PDUE_SIZE];          // Amount past due
   char again;                       // To hold Y or N

   do
   {
      // Call getInfo to get input from the user
      getInfo(salutation, lastName, balance, pastDue,
              lastPayment);
      cout << "\n\n";
      // Now print the form letter
      printLetter(salutation, lastName, balance, pastDue,
                  lastPayment);
      cout << "\n\nDo another letter? (Y/N) ";
      cin >> again;
   } while (toupper(again) == 'Y');
   return 0;
}
```


Notice pointers to `salutation`, `lastName`, `balance`, `pastDue`, and `lastPayment` are passed to `getInfo` and `printLetter`. When `getInfo` returns, these fields will have the values provided by the user stored in them. `printLetter` will retrieve the values and insert them in the form letter.

The `getInfo` Function

This function first calls the `getSal` function (to get the salutation), then asks the user to enter the customer's last name, account balance, past-due amount, and the date of the last payment. These values are then stored in the arrays whose addresses are passed into the function as arguments. Here is the pseudocode:

```
Call getSal.
Ask the user to enter the customer's last name.
Convert the first character of the last name to upper case.
Ask the user to enter the customer's account balance.
Ask the user to enter the account's past-due amount.
Ask the user to enter the date of the last payment.
```

Notice after the user enters the customer's last name, the function automatically converts its first character to upper case. This is in case the user entered the name in all lower case. Here is the function's C++ code:

```
void getInfo(char *sal, char *lname, char *bal, char *due, char *lastPay)
{
   getSal(sal);
   cout << "Last name: ";
   cin >> lname;
   lname[0] = toupper(lname[0]);
   cout << "Account balance: ";
   cin >> bal;
   cout << "Past due Amount: ";
   cin >> due;
   cout << "Date of last payment (MM/DD/YYYY): ";
   cin >> lastPay;
}
```

The `getSal` Function

This function displays a menu allowing the user to select a salutation from either Mr. or Ms. The choice is then stored in the array whose address is passed into the function as an argument. Here is the pseudocode:


```
Do
   Display menu with choice 1 being Mr. and choice 2 being Ms.
   Ask user to select a salutation.
While the user does not select 1 or 2 from the menu.
If the user selected 1
    The salutation is Mr.
else
    The salutation is Ms.
End If.
```

Here is the function's C++ code:

```
void getSal(char *sal)
{
   int choice;

   do
   {
      cout << "salutation:\n";
      cout << "\t1) Mr.\n";
      cout << "\t2) Ms.\n";
      cout << "Select one: ";
      cin >> choice;
   } while (choice != 1 && choice != 2);

   if (choice == 1)
      strcpy(sal, "Mr.");
   else
      strcpy(sal, "Ms.");
}
```

The `printLetter` Function

Once the user has entered values for all the fields, this function controls the printing of the letter. It has one local variable, `position`, which is an integer. This variable keeps track of the number of characters printed on the current line. This is crucial information for the `printLine` function, which performs word-wrap. Below is the function's pseudocode. (It might help you to refer to the contents of the global arrays `part1` through `part8` as you read the code.)

```
// First print the salutation part of the letter.
Set the position variable to 0 (for printLine).
Call printLine to print the part1 array.
Print the salutation, followed by a space, followed by the
customer's last name, followed by a colon.
// Next print the body of the letter.
Set the position variable to zero.
Call printLine to print the part2 array.
Print the customer's balance.
Adjust the position variable.
Call printLine to print the part3 array.
Print the past-due amount.
Adjust the position variable.
Call printLine to print the part4 array.
Print the date of the last payment.
Adjust the position variable.
Call printLine to print the part5 array.
// Next print the letter's closing.
Set the position variable to zero (to start a new line.)
Call printLine to print the part6 array.
Set the position variable to zero (to start a new line).
Call printLine to print the part7 array.
// Last, print the PS reminder.
Set the position variable to zero (to start a new line).
Call printLine to print the part8 array.
```


The `printLine` function updates the `position` variable. When `PrintLetter` prints one of the fields, such as balance, it must adjust the position variable. This is so the `printLine` function will accurately detect the end of each line. Notice every time a new line is to be started, `position` is reset to zero. Here is the C++ code for the function:

```
void printLetter(char *sal, char *lname, char *bal, char *due,
                 char *lastPay)
{
   int position;

   // Print the salutation part of the letter
   position = 0; // Start a new line.
   printline(part1, position);
   cout << sal << " " << lname << ":" << endl << endl;

   // Print the body of the letter
   position = 0; // Start a new line.
   printline(part2, position);
   cout << bal;  // Print account balance.
```

```
    // Add length of balance to position.
    position += strlen(bal);
    printline(part3, position);
    cout << due << ". "; // Print past due amount
    position += strlen(due) + 2;

    // Add length of due and the period and space at the
    // end of the sentence to position.
    printline(part4, position);
    cout << lastPay << ". "; // Print date of last payment.

    // Now Add length of lastPay and the period and space at the
    // end of the sentence to position.
    position += strlen(lastPay) + 2;
    printline(part5, position);

    // Print the closing.
    position = 0; // Start a new line.
    printline(part6, position);
    position = 0; // Start a new line.
    printline(part7, position);

    // Print the PS reminder.
    position = 0; // Start a new line.
    printline(part8, position);
}
```


The `printLine` Function

This function prints each individual line of the letter. It takes two arguments: the address of the string that is to be printed on the line, and the variable used to store the number of characters printed (the `position` variable). The number of characters printed on the line is important because the program must perform word-wrap. This happens when the word being printed at the end of a line will not entirely fit. Instead of printing part of the word on one line then continuing it on the next line, the program is to start the word on the next line. Here is the function's pseudocode:

```
If the line is at or past the right margin
    Start a new line.
End If.
While not at the end of the string
    If 60 or more characters have been printed AND the next
    character is a space
        Perform word-wrap.
    End If.
    If at the beginning of a new line
        Print the left margin (10 spaces).
        Add 10 to the number of characters printed.
    End If.
    Print the next character.
    Add one to the number of characters printed.
End While.
```

The first `if` statement simply checks to see if the current printing position is at or beyond the right margin. Because the letter has ten-character margins, any position beyond the 70th character is in the right margin.

Inside the `while` loop, another `if` statement checks to see if 60 or more characters have been printed. This is the part that controls word-wrap. The function begins watching for a space separating words at the 60th character. If a break between two words appears anywhere after the 60th character, a new line is started. The next `if` statement checks to see `if` a new line has begun. If so, it prints the ten spaces that make the left margin. After all this has taken place, the next character is printed and the character count is incremented.

Here is the C++ code for the function:

```
void printline(const char *line, int &startCount)
{
   const int LMARGIN = 10;
   const int RMARGIN = 70;
   const int WRAP = 60;

   int charCount = 0;

   if (startCount >= RMARGIN) // If the line is already at
   {                          // or past the right margin...
      cout << "\n";           // Start a new line.
      startCount = 0;         // Reset startCount.
   }
   // The following while loop cycles through the string
   // printing it one character at a time. It watches for
   // spaces after the 60th position so word-wrap may be
   // performed.
   while (line[charCount] != '\0')
   {
      if (startCount >= WRAP && line[charCount] == ' ')
      {
         cout << "         \n";  // Print right margin.
         charCount++;            // Skip over the space.
         startCount = 0;
      }
      if (startCount == 0)
      {
         cout << "         ";    // Print left margin.
         startCount = LMARGIN;
      }
      cout.put(line[charCount]); // Print the character.
      charCount++;               // Update subscript.
      startCount++;              // Update position counter.
   }
}
```

NOTE: The `startCount` parameter is a reference to the `position` variable in the `printLetter` function.

The Entire Program

Program CS1-1 shows the entire program's source code.

Program CS1-1

```
1  // This program prints a simple form letter reminding a customer
2  // of an overdue account balance.
3  #include <iostream>
4  #include <cctype>
5  #include <cstring>
6  using namespace std;
7
8  // Function Prototypes
9  void printLetter(char *, char *, char *, char *, char *);
10 void getInfo(char *, char *, char *, char *, char *);
11 void getSal(char *);
12 void printline(const char *, int&);
13
14 // Strings that make up the form letter
15 const char part1[] = "Dear ";
16 const char part2[] = "Our records show that your account has a"
17                      " balance of $";
18 const char part3[] = " and a past-due amount of $";
19 const char part4[] = "Your last payment was on ";
20 const char part5[] = "Since we haven't heard from you in some"
21                      " time, would you please take a moment to send"
22                      " us a check for the past-due amount? We value"
23                      " your business and look forward to serving you"
24                      " in the future.\n\n";
25 const char part6[] = "Sincerely,\n";
26 const char part7[] = "The Management\n\n";
27 const char part8[] = "P.S. If you've already sent your payment, ignore"
28                      " this reminder.";
29
30 int main()
31 {
32    // Constants for array sizes
33    const int SAL_SIZE = 4;           // salutation size
34    const int LNAME_SIZE = 16;        // lastName size
35    const int LPAYMENT_SIZE = 16;     // lastPayment size
36    const int BAL_SIZE = 9;           // balance size
37    const int PDUE_SIZE = 9;          // pastDue size
38
```

(program continues)

Program CS1-1 *(continued)*

```
39      char salutation[SAL_SIZE];         // To hold the salutation
40      char lastName[LNAME_SIZE];         // Customer's last name
41      char lastPayment[LPAYMENT_SIZE];   // Date of last payment
42      char balance[BAL_SIZE];            // Account balance
43      char pastDue[PDUE_SIZE];           // Amount past due
44      char again;                        // To hold Y or N
45
46      do
47      {
48         // Call getInfo to get input from the user
49         getInfo(salutation, lastName, balance, pastDue,
50                 lastPayment);
51         cout << "\n\n";
52
53         // Now print the form letter
54         printLetter(salutation, lastName, balance, pastDue,
55                     lastPayment);
56         cout << "\n\nDo another letter? (Y/N) ";
57         cin >> again;
58      } while (toupper(again) == 'Y');
59      return 0;
60  }
61
62  //*************************************************************
63  // Definition of function getInfo.                            *
64  // This function allows the user to enter the following items:*
65  // salutation, last name, account balance, past due amount, and*
66  // date of last payment. The function arguments are pointers to*
67  // strings where the input will be stored.                    *
68  //*************************************************************
69
70  void getInfo(char *sal, char *lname, char *bal, char *due, char *lastPay)
71  {
72     getSal(sal);
73     cout << "Last name: ";
74     cin >> lname;
75     lname[0] = toupper(lname[0]);
76     cout << "Account balance: ";
77     cin >> bal;
78     cout << "Past due Amount: ";
79     cin >> due;
80     cout << "Date of last payment (MM/DD/YYYY): ";
81     cin >> lastPay;
82  }
```

```
83
84  //*************************************************************
85  // Definition of function getSal.                             *
86  // This function gives the user a menu from which to pick a   *
87  // suitable title for the letter's addressee. The choices are *
88  // Mr. and Ms. The choice will be copied to the address pointed*
89  // to by sal.                                                 *
90  //*************************************************************
91
92  void getSal(char *sal)
93  {
94     int choice;
95
96     do
97     {
98        cout << "salutation:\n";
99        cout << "\t1) Mr.\n";
100       cout << "\t2) Ms.\n";
101       cout << "Select one: ";
102       cin >> choice;
103    } while (choice != 1 && choice != 2);
104
105    if (choice == 1)
106       strcpy(sal, "Mr.");
107    else
108       strcpy(sal, "Ms.");
109 }
110
111 //*************************************************************
112 // Definition of function printLetter.                        *
113 // This function prints the form letter. The parameters are   *
114 // pointers to the fields that contain user input.            *
115 //*************************************************************
116
117 void printLetter(char *sal, char *lname, char *bal, char *due,
118                  char *lastPay)
119 {
120    int position;
121
122    // Print the salutation part of the letter
123    position = 0; // Start a new line.
124    printline(part1, position);
125    cout << sal << " " << lname << ":" << endl << endl;
126
```


(program continues)

Program CS1-1 *(continued)*

```
    // Print the body of the letter
    position = 0; // Start a new line.
    printline(part2, position);
    cout << bal;  // Print account balance.

    // Add length of balance to position.
    position += strlen(bal);
    printline(part3, position);
    cout << due << ". "; // Print past due amount
    position += strlen(due) + 2;

   // Add length of due and the period and space at the
   // end of the sentence to position.
   printline(part4, position);
   cout << lastPay << ". "; // Print date of last payment.

   // Now Add length of lastPay and the period and space at the
   // end of the sentence to position.
   position += strlen(lastPay) + 2;
   printline(part5, position);

   // Print the closing.
   position = 0; // Start a new line.
   printline(part6, position);
   position = 0; // Start a new line.
   printline(part7, position);

   // Print the PS reminder.
   position = 0; // Start a new line.
   printline(part8, position);
}

//*********************************************************************
// Definition of function printline.                                  *
// This function has two parameters: line and startCount.             *
// The string pointed to by line is printed. startCount is the        *
// starting position of the line in an 80 character field. There      *
// are 10-character left and right margins within the 80              *
// character field. The function performs word-wrap by looking        *
// for space character within the line at or after the 60th           *
// character. A new line is started when a space is found, or the     *
// end of the field is reached.                                       *
//*********************************************************************
```

```
171  void printline(const char *line, int &startCount)
172  {
173     const int LMARGIN = 10;
174     const int RMARGIN = 70;
175     const int WRAP = 60;
176
177     int charCount = 0;
178
179     if (startCount >= RMARGIN)    // If the line is already at
180     {                             // or past the right margin...
181        cout << "\n";              // Start a new line.
182        startCount = 0;            // Reset startCount.
183     }
184
185     // The following while loop cycles through the string
186     // printing it one character at a time. It watches for
187     // spaces after the 60th position so word-wrap may be
188     // performed.
189     while (line[charCount] != '\0')
190     {
191        if (startCount >= WRAP && line[charCount] == ' ')
192        {
193           cout << "\n";           // Print right margin.
194           charCount++;            // Skip over the space.
195           startCount = 0;
196        }
197        if (startCount == 0)
198        {
199           cout << "          ";  // Print left margin.
200           startCount = LMARGIN;
201        }
202        cout.put(line[charCount]); // Print the character.
203        charCount++;               // Update subscript.
204        startCount++;              // Update position counter.
205     }
206  }
```


Program Output with Example Input Shown in Bold

```
Salutation:
      1) Mr.
      2) Ms.
Select one: 1 Enter
Last name: Jones Enter
Account balance: 267.98 Enter
Past due amount: 57.13 Enter
Date of last payment(MM/DD/YYYY): 2/14/2018 Enter
```

(program output continues)

Program CS1-1 *(continued)*

```
Dear Mr. Jones:

Our records show that your account has a balance of
$267.98 and a past-due amount of $57.13. Your last
payment was on 2/14/2018. Since we haven't heard from
you in some time, would you please take a moment to
send us a check for the past-due amount? We value your business
and look forward to serving you in the future.

Sincerely,
The Management

P.S. If you've already sent your payment, ignore this
reminder.

Do another letter? (Y/N) y Enter
Salutation:
      1) Mr.
      2) Ms.
Select one: 2 Enter
Last name: Hildebrand Enter
Account balance: 4,598.00 Enter
Past due amount: 1,367.00 Enter
Date of last payment(MM/DD/YYYY): 5/23/2018 Enter

Dear Ms. Hildebrand:

Our records show that your account has a balance of
$4,598.00 and a past-due amount of $1,367.00. Your
last payment was on 5/23/2018. Since we haven't heard
from you in some time, would you please take a moment
to send us a check for the past-due amount? We value
your business and look forward to serving you in the
future.

Sincerely,
The Management

P.S. If you've already sent your payment, ignore this
reminder.

Do another letter? (Y/N) n Enter
```