```
2 * PROGRAMMED BY : Nick Reardon
3 * CLASS
                 : CS1D
                 : MW - 2:30p
4 * SECTION
 5 * Assignment #1 : Recursion
   *****************************
7
8
                           Assignment #1 - Recursion
9
10
      Write a recursive function that takes a string as an argument and
11 returns a TRUE if the string is a palindrome otherwise FALSE is
   returned. A palindrome is a string that reads the same forward
   and backward (dad, mom, radar). Test with a main program that
13
14 calls the function. The main function could solicit for input and
15 terminate after receiving an EOF or one can store the strings in a
16 vector or array.
17
18
       The recursive function should ignore white spaces, capital letters
19
   (case insensitive), and punctuation. For example: ôA man a plan a
20 canal Panamaö would be a palindrome.
21
22
       -Use paragraphing and comment each logical block of source code.
23
       -Do not use global variables. Your output should be descriptive.
24
25 Use the following input:
26 A man a plan a canal Panama
   The rain in Spain
28 No lemon, no melon
29 radar
30 CS1D
31 Was it a cat I saw?
32 Racecar
33 Saddleback
34 dad
35
36 Due January 22nd
   ***************************
37
38
39
      ** Populating vector with strings read from file
40
      ** Testing each string in the vector for palindrome
41
42
43
44
    | A man a plan a canal Panama |
                                 MATCH
45
46
    | A man a plan a canal Panama |
                                 MATCH
47
48
    | A man a plan a canal Panama |
                                 MATCH
49
50
    | A man a plan a canal Panama |
                                 MATCH
51
52
    A man a plan a canal Panama
                                 MATCH
```

```
54
     | A man a plan a canal Panama | MATCH
55
56
     A man a plan a canal Panama
                                      MATCH
57
58
     | A man a plan a canal Panama | MATCH
59
60
     A man a plan a canal Panama | MATCH
61
62
     | A man a plan a canal Panama | MATCH
63
64
     | A man a plan a canal Panama | MATCH
65
66
     This IS a palindrome
67
68
69
70
     | The rain in Spain | NO MATCH
71
72
     This is NOT a palindrome
73
74
75
76
     No lemon, no melon | MATCH
77
78
     | No lemon, no melon | MATCH
79
80
     | No lemon, no melon | MATCH
81
82
     | No lemon, no melon | MATCH
83
84
     | No lemon, no melon | MATCH
85
86
     | No lemon, no melon | MATCH
87
     | No lemon, no melon | MATCH
88
89
90
     This IS a palindrome
91
93
     | radar | MATCH
94
95
     | radar | MATCH
96
        ^ ^
97
     | radar | MATCH
98
99
100
     This IS a palindrome
101
102
103
     | CS1D | NO MATCH
104
```

```
105
106
    This is NOT a palindrome
107
108 -----
109
     | Was it a cat I saw? | MATCH
110
111
     | Was it a cat I saw? | MATCH
112
113
114
     | Was it a cat I saw? | MATCH
115
116
     | Was it a cat I saw? | MATCH
     ^ ^
117
118
     | Was it a cat I saw? | MATCH
119
120
     | Was it a cat I saw? | MATCH
121
122
     | Was it a cat I saw? | MATCH
123
124
    This IS a palindrome
125
126 -----
127
128
   | Racecar | MATCH
129
     ^ ^
130
     | Racecar | MATCH
      ^ ^
131
     | Racecar | MATCH
132
133
      ^ ^
134
     | Racecar | MATCH
135
136
    This IS a palindrome
137
138 -----
139
     | Saddleback | NO MATCH
140
141
142
    This is NOT a palindrome
143
144 -----
145
146
     | dad | MATCH
     ^ ^
147
     | dad | MATCH
148
149
150
    This IS a palindrome
151
152 -----
154 Press any key to continue . . .
```

```
2
   * AUTHOR
                    : Nick Reardon
 3 * Assignment #1
                   : Vectors
 4 * CLASS
                     : CS1D
 5
   * SECTION
                     : MW - 2:30p
   * DUE DATE
 6
                     : 01 / 22 / 20
   *******************************
 8 #ifndef _MAIN_H_
9 #define _MAIN_H_
10
11 //Standard includes
12 #include <iostream>
13 #include <iomanip>
14 #include <string>
15 #include "PrintHeader.h"
17 //Program Specific
18 #include <fstream>
19 #include <vector>
20
21
22 // Setup function for PalindromeRecursion
23 // returns true if the given string IS a palindrome, else it returns false
24 // No change to given string
25 bool CheckPalindrome(const std::string& input);
26
27
28 // Recursively checks if a given string is a palindrome
29 // Uses setup function CheckPalindrome
30 // returns true if the given string IS a palindrome, else it returns false
31 // No change to given string
32 // Case insensitive, ignores whitespace and any non alpha numeric character
33 bool PalindromeRecursion(const std::string& input, int front, int back);
34
35
36
37 // Outputs a given string along with given index locations
38 // Used to indicate current character comparisons for palindromes
39 // Indicates two given indices unless both indices match
40 // No change to given string
41 void PrintStringPositions(const std::string& input, int front, int back);
42
43 #endif // _HEADER_H_
44
```

```
...urce\repos\CS1D-AS1-Recursion\CS1D AS1 Recursion\main.cpp
```

```
-
```

```
2
   * AUTHOR
                   : Nick Reardon
3
   * Assignment #1
                 : Vectors
4
   * CLASS
                   : CS1D
5
   * SECTION
                   : MW - 2:30p
6
   * DUE DATE
                   : 01 / 22 / 20
   7
8 #include "main.h"
10 using std::cout; using std::endl;
11
12
13 int main()
14 {
15
16
       * HEADER OUTPUT
17
18
19
      PrintHeader(cout, "Prompt.txt");
20
      21
22
23
      // input file variable setup
24
      std::ifstream iFile;
25
      iFile.open("Input.txt");
26
27
      std::vector<std::string> inputVect;
      cout << " ** Populating vector with strings read from file" << endl;</pre>
28
29
30
      // reading from input file into vector
31
      while (iFile)
32
      {
33
         std::string temp;
34
         std::getline(iFile, temp);
35
         inputVect.push_back(temp);
36
      }
37
38
      cout << " ** Testing each string in the vector for palindrome" << endl;</pre>
39
40
      //Checking each string for palindromes in a loop by calling CheckPalindrome >
        function
41
      for (int i = 0; i < inputVect.size() - 1; i++)</pre>
42
         cout << endl << "-----" << endl << endl;</pre>
43
44
45
         if (CheckPalindrome(inputVect[i]))
46
47
             cout << " This IS a palindrome" << endl;</pre>
48
         }
49
         else
50
         {
             cout << " This is NOT a palindrome" << endl;</pre>
51
```

```
52
53
            }
54
        }
        cout << endl << "----" << endl << endl;</pre>
55
56
57
58
        system("pause");
59
        return 0;
60 }
61
62
63 // Setup function for PalindromeRecursion
64 // returns true if the given string IS a palindrome, else it returns false
65 // No change to given string
66 bool CheckPalindrome(const std::string& input)
67 {
68
        int back = input.length() - 1;
69
70
        return PalindromeRecursion(input, 0, back);
71 }
72
73
74 // Recursively checks if a given string is a palindrome
75 // Uses setup function CheckPalindrome
76 // returns true if the given string IS a palindrome, else it returns false
77 // No change to given string
78 // Case insensitive, ignores whitespace and any non alpha numeric character
79 bool PalindromeRecursion(const std::string& input, int front, int back)
80 {
81
        bool match;
82
        bool validChar = false;
83
        while (validChar == false)
84
85
        {
86
            if (input[front] < '0' ||</pre>
                 (input[front] > '9' && input[front] < 'A') ||
87
                 (input[front] > 'Z' && input[front] < 'a') ||</pre>
88
89
                 input[front] > 'z')
90
            {
91
                 front++;
92
             }
93
            else
94
            {
95
                 validChar = true;
96
97
98
            if (input[back] < '0' ||</pre>
99
                 (input[back] > '9' && input[back] < 'A') ||
                 (input[back] > 'Z' && input[back] < 'a') ||</pre>
100
101
                 input[back] > 'z')
102
            {
103
                 back--;
```

```
104
                 validChar = false;
105
106
             }
107
             else
108
             {
109
                 validChar = true;
110
             }
111
112
             if (!validChar && back <= front)</pre>
113
114
                 return true;
115
             }
116
         }
117
118
         PrintStringPositions(input, front, back);
119
120
         if (toupper(input[front]) != toupper(input[back]))
121
         {
122
             return false;
123
124
         else if ((back - front) < 2)</pre>
125
126
             return true;
127
         }
128
         else
129
130
             return PalindromeRecursion(input, ++front, --back);
131
132 }
133
134
135 // Outputs a given string along with given index locations
136 // Used to indicate current character comparisons for palindromes
137 // Indicates two given indices unless both indices match
138 // No change to given string
139 void PrintStringPositions(const std::string& input, int front, int back)
140 {
141
142
         if ((front - back) == 0)
143
             cout << " | " << input << " | ";</pre>
144
             if (toupper(input[front]) == toupper(input[back]))
145
146
             {
                 cout << " MATCH" << endl;</pre>
147
             }
148
149
             else
150
             {
151
                 cout << " NO MATCH" << endl;</pre>
152
153
             cout << " " << std::string(front, ' ') << '^' << endl;</pre>
154
         }
155
         else
```

```
\dotsurce\repos\CS1D-AS1-Recursion\CS1D AS1 Recursion\main.cpp
```

```
156
          cout << " | " << input << " | ";</pre>
157
          if (toupper(input[front]) == toupper(input[back]))
158
159
             cout << " MATCH" << endl;</pre>
160
161
          }
162
          else
163
          {
             cout << " NO MATCH" << endl;</pre>
164
165
          166
167
168
       }
169
170
171 }
```

```
: Nick Reardon
  * AUTHOR
3 * Assignment #1 : Vectors
4 * CLASS
               : CS1D
5 * SECTION
                : MW - 2:30p
  * DUE DATE
               : 01 / 22 / 20
  ********************************
8 #ifndef _PRINTHEADER_H_
9 #define _PRINTHEADER_H_
10
11 #include <iostream>
12 #include <iomanip>
13 #include <ostream>
14 #include <string>
15 #include <fstream>
18 * PrintHeader
19 * -----
20 * This function will output a class header through the use of ostream.
21 * It also will output the program description
23 * Call
24 * -----
25 * The function call requires 1 parameters. The following example uses an
26 * output file in the ostream parameter. Ex:
27 *
28 *
        PrintHeader (oFile);
29 *
30 * -----
31 * Output
33 * The function will output as follows. Ex:
34 *
       *****************
35 *
36 *
       * PROGRAMMED BY : Parsa Khazravi and Nick Reardon
37 *
       * CLASS : CS1B
38 *
                : MW: 7:30pm
: Functions - GCD
       * SECTION
       * Lab #3
       ****************
40 *
41 *
43 * CONSTANTS
45 * OUTPUT - USED FOR CLASS HEADING
46 * ------
               : Name(s) of programmer(s) - Nick Reardon
47 * PROGRAMMER
                                 - MW - 7:30p
48 * SECTION
               : Class times
49 * CLASS
                : Class label
                                   - CS1B
             : # of the program
50 * PROGRAM_NUM
             : Title of the program: Type of program - Lab, Assignment, etc.
51 * PROGRAM NAME
52 * PROGRAM TYPE
```

```
53 *
54 * -----
55 * MAX OUTPUT : Max movies to be output at once
57 const std::string PROGRAMMER = "Nick Reardon";
58 const std::string SECTION = "MW - 2:30p";
59 const std::string CLASS = "CS1D";
60 const int PROGRAM NUM = 1;
61 const std::string PROGRAM NAME = "Recursion";
62 const std::string PROGRAM_TYPE = "Assignment";
63
64
65 void PrintHeader(std::ostream &output, std::string inputText)
66 {
67
       std::string typeNum = PROGRAM_TYPE + " #" + std::to_string(PROGRAM_NUM);
68
      output << std::left</pre>
69
          << std::string(76, '*')
71
          << std::endl
          << "* PROGRAMMED BY : " << PROGRAMMER << std::endl</pre>
72
          << "* " << std::setw(14) << "CLASS" << ": " << CLASS << std::endl</pre>
73
          << "* " << std::setw(14) << "SECTION" << ": " << SECTION << std::endl</pre>
74
          << "* " << std::setw(14) << typeNum << ": " << PROGRAM_NAME << std::endl</pre>
75
76
          << std::string(76, '*')
77
          << std::endl << std::endl</pre>
78
          << std::string(((76 - typeNum.length() - PROGRAM_NAME.length() ) / 2), ' >
            ')
79
          << typeNum + " - " + PROGRAM_NAME
          << std::endl << std::endl
          << std::ifstream(inputText).rdbuf()</pre>
81
          << std::endl</pre>
82
         << std::string(76, '*')</pre>
83
         << std::endl << std::endl;</pre>
84
85
86 }
87
88 #endif //_PRINTHEADER_H_
```