```cpp
/*****************************************************************************
 * AUTHOR           : Nick Reardon
 * Assignment #1    : Vectors
 * CLASS            : CS1D
 * SECTION          : MW - 2:30p
 * DUE DATE         : 01 / 22 / 20
 *****************************************************************************/
#include "main.h"

using std::cout; using std::endl;


int main()
{

    /*
     * HEADER OUTPUT
     */
    PrintHeader(cout, "Prompt.txt");

    /*****************************************************************/

    // input file variable setup
    std::ifstream iFile;
    iFile.open("Input.txt");

    std::vector<std::string> inputVect;
    cout << "   ** Populating vector with strings read from file" << endl;

    // reading from input file into vector
    while (iFile)
    {
        std::string temp;
        std::getline(iFile, temp);
        inputVect.push_back(temp);
    }

    cout << "   ** Testing each string in the vector for palindrome" << endl;

    //Checking each string for palindromes in a loop by calling CheckPalindrome function
    for (int i = 0; i < inputVect.size() - 1; i++)
    {
        cout << endl << "------------------------" << endl << endl;

        if (CheckPalindrome(inputVect[i]))
        {
            cout << " This IS a palindrome" << endl;
        }
        else
        {
            cout << " This is NOT a palindrome" << endl;
```

```
52
53          }
54      }
55      cout << endl << "-------------------------" << endl << endl;
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
84      while (validChar == false)
85      {
86          if (input[front] < '0' ||
87              (input[front] > '9' && input[front] < 'A') ||
88              (input[front] > 'Z' && input[front] < 'a') ||
89              input[front] > 'z')
90          {
91              front++;
92          }
93          else
94          {
95              validChar = true;
96          }
97
98          if (input[back] < '0' ||
99              (input[back] > '9' && input[back] < 'A') ||
100             (input[back] > 'Z' && input[back] < 'a') ||
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)
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)
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     {
144         cout << " | " << input << " | ";
145         if (toupper(input[front]) == toupper(input[back]))
146         {
147             cout << " MATCH" << endl;
148         }
149         else
150         {
151             cout << " NO MATCH" << endl;
152         }
153         cout << "   " << std::string(front, ' ') << '^' << endl;
154     }
155     else
```

```
156     {
157         cout << " | " << input << " | ";
158         if (toupper(input[front]) == toupper(input[back]))
159         {
160             cout << " MATCH" << endl;
161         }
162         else
163         {
164             cout << " NO MATCH" << endl;
165         }
166         cout << "   " << std::string(front, ' ') << '^'
167             << std::string((back - front) - 1, ' ') << '^' << endl;
168     }
169 
170 
171 }
```