```
****************************************************************************
* PROGRAMMED BY : Nick Reardon
* CLASS         : CS1D
* SECTION       : MW - 2:30p
* Assignment #1 : Recursion
****************************************************************************

                           Assignment #1 - Recursion

    Write a recursive function that takes a string as an argument and
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
calls the function.  The main function could solicit for input and
terminate after receiving an EOF or one can store the strings in a
vector or array.

    The recursive function should ignore white spaces, capital letters
(case insensitive), and punctuation. For example: ôA man a plan a
canal Panamaö would be a palindrome.

    -Use paragraphing and comment each logical block of source code.
    -Do not use global variables. Your output should be descriptive.

Use the following input:
A man a plan a canal Panama
The rain in Spain
No lemon, no melon
radar
CS1D
Was it a cat I saw?
Racecar
Saddleback
dad

Due January 22nd
****************************************************************************

   ** Populating vector with strings read from file
   ** Testing each string in the vector for palindrome

-------------------------

 | A man a plan a canal Panama |  MATCH
   ^                          ^
 | A man a plan a canal Panama |  MATCH
     ^                      ^
 | A man a plan a canal Panama |  MATCH
      ^                    ^
 | A man a plan a canal Panama |  MATCH
       ^                  ^
 | A man a plan a canal Panama |  MATCH
```

```
         ^               ^
 | A man a plan a canal Panama |  MATCH
           ^            ^
 | A man a plan a canal Panama |  MATCH
            ^         ^
 | A man a plan a canal Panama |  MATCH
             ^       ^
 | A man a plan a canal Panama |  MATCH
              ^     ^
 | A man a plan a canal Panama |  MATCH
                ^  ^
 | A man a plan a canal Panama |  MATCH
                  ^
 This IS a palindrome

-------------------------

 | The rain in Spain |  NO MATCH
   ^               ^
 This is NOT a palindrome

-------------------------

 | No lemon, no melon |  MATCH
   ^                ^
 | No lemon, no melon |  MATCH
    ^              ^
 | No lemon, no melon |  MATCH
      ^           ^
 | No lemon, no melon |  MATCH
       ^         ^
 | No lemon, no melon |  MATCH
        ^       ^
 | No lemon, no melon |  MATCH
         ^    ^
 | No lemon, no melon |  MATCH
          ^  ^
 This IS a palindrome

-------------------------

 | radar |  MATCH
   ^   ^
 | radar |  MATCH
    ^ ^
 | radar |  MATCH
     ^
 This IS a palindrome

-------------------------

 | CS1D |  NO MATCH
```

```
   ^  ^
 This is NOT a palindrome

-------------------------

 | Was it a cat I saw? |  MATCH
   ^                ^
 | Was it a cat I saw? |  MATCH
    ^              ^
 | Was it a cat I saw? |  MATCH
     ^            ^
 | Was it a cat I saw? |  MATCH
       ^        ^
 | Was it a cat I saw? |  MATCH
        ^     ^
 | Was it a cat I saw? |  MATCH
          ^  ^
 | Was it a cat I saw? |  MATCH
            ^
 This IS a palindrome

-------------------------

 | Racecar |  MATCH
   ^     ^
 | Racecar |  MATCH
    ^   ^
 | Racecar |  MATCH
     ^ ^
 | Racecar |  MATCH
      ^
 This IS a palindrome

-------------------------

 | Saddleback |  NO MATCH
   ^        ^
 This is NOT a palindrome

-------------------------

 | dad |  MATCH
   ^ ^
 | dad |  MATCH
    ^
 This IS a palindrome

-------------------------

Press any key to continue . . .
```

```cpp
/*****************************************************************************
 * AUTHOR           : Nick Reardon
 * Assignment #1    : Vectors
 * CLASS            : CS1D
 * SECTION          : MW - 2:30p
 * DUE DATE         : 01 / 22 / 20
 *****************************************************************************/
#ifndef _MAIN_H_
#define _MAIN_H_

 //Standard includes
#include <iostream>
#include <iomanip>
#include <string>
#include "PrintHeader.h"

//Program Specific
#include <fstream>
#include <vector>


// Setup function for PalindromeRecursion
// returns true if the given string IS a palindrome, else it returns false
// No change to given string
bool CheckPalindrome(const std::string& input);


// Recursively checks if a given string is a palindrome
// Uses setup function CheckPalindrome
// returns true if the given string IS a palindrome, else it returns false
// No change to given string
// Case insensitive, ignores whitespace and any non alpha numeric character
bool PalindromeRecursion(const std::string& input, int front, int back);



// Outputs a given string along with given index locations
// Used to indicate current character comparisons for palindromes
// Indicates two given indices unless both indices match
// No change to given string
void PrintStringPositions(const std::string& input, int front, int back);

#endif // _HEADER_H_

```

```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;

        }
    }
    cout << endl << "-------------------------" << endl << endl;


    system("pause");
    return 0;
}


// Setup function for PalindromeRecursion
// returns true if the given string IS a palindrome, else it returns false
// No change to given string
bool CheckPalindrome(const std::string& input)
{
    int back = input.length() - 1;

    return PalindromeRecursion(input, 0, back);
}


// Recursively checks if a given string is a palindrome
// Uses setup function CheckPalindrome
// returns true if the given string IS a palindrome, else it returns false
// No change to given string
// Case insensitive, ignores whitespace and any non alpha numeric character
bool PalindromeRecursion(const std::string& input, int front, int back)
{
    bool match;
    bool validChar = false;

    while (validChar == false)
    {
        if (input[front] < '0' ||
            (input[front] > '9' && input[front] < 'A') ||
            (input[front] > 'Z' && input[front] < 'a') ||
            input[front] > 'z')
        {
            front++;
        }
        else
        {
            validChar = true;
        }

        if (input[back] < '0' ||
            (input[back] > '9' && input[back] < 'A') ||
            (input[back] > 'Z' && input[back] < 'a') ||
            input[back] > 'z')
        {
            back--;
            validChar = false;

        }
        else
        {
            validChar = true;
        }

        if (!validChar && back <= front)
        {
            return true;
        }
    }

    PrintStringPositions(input, front, back);

    if (toupper(input[front]) != toupper(input[back]))
    {
        return false;
    }
    else if ((back - front) < 2)
    {
        return true;
    }
    else
    {
        return PalindromeRecursion(input, ++front, --back);
    }
}


// Outputs a given string along with given index locations
// Used to indicate current character comparisons for palindromes
// Indicates two given indices unless both indices match
// No change to given string
void PrintStringPositions(const std::string& input, int front, int back)
{

    if ((front - back) == 0)
    {
        cout << " | " << input << " | ";
        if (toupper(input[front]) == toupper(input[back]))
        {
            cout << " MATCH" << endl;
        }
        else
        {
            cout << " NO MATCH" << endl;
        }
        cout << "   " << std::string(front, ' ') << '^' << endl;
    }
    else
    {
        cout << " | " << input << " | ";
        if (toupper(input[front]) == toupper(input[back]))
        {
            cout << " MATCH" << endl;
        }
        else
        {
            cout << " NO MATCH" << endl;
        }
        cout << "   " << std::string(front, ' ') << '^'
            << std::string((back - front) - 1, ' ') << '^' << endl;
    }


}
```

```
/*****************************************************************************
 * AUTHOR           : Nick Reardon
 * Assignment #1    : Vectors
 * CLASS            : CS1D
 * SECTION          : MW - 2:30p
 * DUE DATE         : 01 / 22 / 20
 *****************************************************************************/
#ifndef _PRINTHEADER_H_
#define _PRINTHEADER_H_

#include <iostream>
#include <iomanip>
#include <ostream>
#include <string>
#include <fstream>

/*****************************************************************************
* PrintHeader
* ----------------------------------------------------------------------------
* This function will output a class header through the use of ostream.
* It also will output the program description
* -----------------------------------------------------------------------------
*  Call
* ------
* The function call requires 1 parameters. The following example uses an
* output file in the ostream parameter. Ex:
*
*       PrintHeader (oFile);
*
* ----------------------------------------------------------------------------
*  Output
* --------
*   The function will output as follows. Ex:
*
*       **********************************************************
*       * PROGRAMMED BY : Parsa Khazravi and Nick Reardon
*       * CLASS         : CS1B
*       * SECTION       : MW: 7:30pm
*       * Lab #3        : Functions - GCD
*       *********************************************************
*
* ----------------------------------------------------------------------------
* CONSTANTS
* ----------------------------------------------------------------------------
* OUTPUT - USED FOR CLASS HEADING
* ----------------------------------------------------------------------------
* PROGRAMMER        : Name(s) of programmer(s)  - Nick Reardon
* SECTION           : Class times               - MW - 7:30p
* CLASS             : Class label               - CS1B
* PROGRAM_NUM       : # of the program
* PROGRAM_NAME      : Title of the program
* PROGRAM_TYPE      : Type of program - Lab, Assignment, etc.
```

```
*
* ---------------------------------------------------------------------------
* MAX_OUTPUT        : Max movies to be output at once
*****************************************************************************/
const std::string PROGRAMMER = "Nick Reardon";
const std::string SECTION = "MW - 2:30p";
const std::string CLASS = "CS1D";
const int PROGRAM_NUM = 1;
const std::string PROGRAM_NAME = "Recursion";
const std::string PROGRAM_TYPE = "Assignment";


void PrintHeader(std::ostream &output, std::string inputText)
{
    std::string typeNum = PROGRAM_TYPE + " #" + std::to_string(PROGRAM_NUM);

    output << std::left
        << std::string(76, '*')
        << std::endl
        << "* PROGRAMMED BY : " << PROGRAMMER << std::endl
        << "* " << std::setw(14) << "CLASS" << ": " << CLASS << std::endl
        << "* " << std::setw(14) << "SECTION" << ": " << SECTION << std::endl
        << "* " << std::setw(14) << typeNum << ": " << PROGRAM_NAME << std::endl
        << std::string(76, '*')
        << std::endl << std::endl
        << std::string(((76 - typeNum.length() - PROGRAM_NAME.length() ) / 2), ' ')
        << typeNum + " - " + PROGRAM_NAME
        << std::endl << std::endl
        << std::ifstream(inputText).rdbuf()
        << std::endl
        << std::string(76, '*')
        << std::endl << std::endl;

}

#endif //_PRINTHEADER_H_
```