1DV533 STEP 4a and 4b Assignment report

Jesper Malmberg em222vs@student.lnu.se

Step 4a

Task 2

In this program I started with 2 constants, max difficulty and max score. I then check that the entered difficulty entered by the user is correct before proceeding. I then make an entry from each judge, again checking that the input is within the required range. Finally I sort the resulting scores in an ascending order and calculate the average score excluding the highest and lowest. I Elected to have to functions in this program named sortArray and calculateScore.

```
//--------------------------------------------------------------------------
//--------------------------------------------------------------------------
// File: Task_2.cpp
// Summary: This program calculatest the score of a dive
// Version: 1.1
// Owner: Jesper Malmberg
//--------------------------------------------------------------------------
// Log: 2021-12-19 Created file
//--------------------------------------------------------------------------
// Preprocessor directives
#include <iostream>

using namespace std;
// Prototypes
void sortArray(int size, double arr[]);
double calculateScore(double arr[], double level);

int main()
{
	// Constants
	const double MAXDIFFICULTY = 4.0;
	const double MAXSCORE = 10;

	double score[7];		// Array to store the scores
	double difficulty;		// The level of difficulty entered

	bool diffOk = false;		// Is correct value entered
	bool scoreOk;			// Is correct score entered

	do {
		cout << "Enter the degree of difficulty for the dive (1.0 - 4.0) : ";

		cin >> difficulty;

		if (difficulty <= MAXDIFFICULTY) {
			diffOk = true;
		}
		if (!diffOk)
			cout << "Try again!" << endl;
		// Keep asking to enter level of difficulty until a correct value is
entered
```

```
	} while (!diffOk);

	// Add the individual judges' scores
	for (int i = 0; i < 7; i++) {
		double temp;
		scoreOk = false;

		do {
			cout << "Enter score for judge " << i + 1 << " (0 - 10) : ";
			cin >> temp;
			if (temp >= 0 && temp <= MAXSCORE) {
				score[i] = temp;

				scoreOk = true;
			}
			if (!scoreOk) {
				cout << "Try again!" << endl;
			}
			// Keep asking to enter score until correct value entered
		} while (!scoreOk);
	}

	// Sort array and print to terminal
	sortArray(7, score);
	cout << endl;
	cout << "The diver's final score is: " << calculateScore(score, difficulty)
<< endl;

	return 0;
}

//---------------------------------------------------------------------
// void sortArray(int size, double arr[])
// Sorts the specified array in ascending order
//
// int size - the size of the array
// double arr[] - the specified array
//---------------------------------------------------------------------
void sortArray(int size, double arr[]) {
	double temp; // Temporary score
	int i, j;
	for (i = 0; i < size; i++) {
		for (j = 0; j < size - 1; j++) {
			if (arr[j] > arr[i]) {
				temp = arr[i]; // swap the scores
				arr[i] = arr[j];
				arr[j] = temp;
			}
		}
	}
}

//---------------------------------------------------------------------
// double calculateScore(double arr[], double level)
// Calculates the score as follows; largest and smallest score disregarded
// the sum of the others are added together, multiplied by level and 0.6
//
// double arr[] - the specified array
```

```
// double level - the specified level of dificulty
//-----------------------------------------------------------------------
double calculateScore(double arr[], double level) {
	double sumOfJudges = 0;
	// Take only the mid 5 numbers ie not index 0 or 6
	for (int i = 1; i < 6; i++) {
		sumOfJudges += arr[i];
	}
	return sumOfJudges * level * 0.6;
}
```

Task 3

For this task I chose to store and represent each fraction in a structure Fraction which simply contains 2 integers, numerator and denominator. The user simply enters the numerator followed by the denominator. The gcdCalculator first checks if the fractional number entered is negative and changes the number to positive in order to find gcd. It then checks if the % operator is equal to 0 for both numerator and denominator and if so that's the gcd which is returned. The GCD is then used to calculate the simplified fractional number.

```
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
// File: Task_3.cpp
// Summary: This program returns lowest common denominator
//
// Version: 1.1
// Owner: Jesper Malmberg
//-----------------------------------------------------------------------
// Log: 2021-12-19 Created file
//-----------------------------------------------------------------------
// Preprocessor directives
#include <iostream>
#include <iomanip>

using namespace std;
// Function prototypes
int gcdCalculator(int num1, int num2);

// The datastructure fraction holds information on the numerator and denominator
struct Fraction {
	int numerator;
	int denominator;
};

int main()
{
	char answer;		// Continue or stop program variable
	Fraction f;			// The fractional number to be calculated


	do {
		printf("\033c");	// Resets the terminal window

		cout << "FRACTION CALCULATION" << endl;
		cout << "====================" << endl;
		cout << endl;
```

```
		cout << "Enter the numerator : ";
		cin >> f.numerator;
		cout << "Enter the denominator : ";
		cin >> f.denominator;

		// If denominator is < 0
		if (f.denominator < 0) {
			f.numerator = -f.numerator;
			f.denominator = -f.denominator;
		}

		// Calculate the gcd
		int gcd = gcdCalculator(f.numerator, f.denominator);

		// Calculate the new fractional numbers base on gcd
		int n = f.numerator / gcd;
		int d = f.denominator / gcd;

		// the whole number plus remaining fraction is found by using %
operator.
		cout << " The fraction can be abbreviated to : " << n << "/" << d << "
= " << n / d << " " << n % d << "/" << d << endl;

		cout << "One more time? (Y/N) ";
		cin >> answer;
	} while (toupper(answer) == 'Y');
	return 0;
}

//---------------------------------------------------------------------------
// gcdCalculator()
// This helper function takes two integers and calculates the greatest common
// denominator. It returns the gcd or 1 if there is none.
//---------------------------------------------------------------------------
int gcdCalculator(int num1, int num2) {

	// If the fractional number is negative change it to positive in order to
find gcd
	if (num1 < 0) {
		num1 = -num1;
	}
	if (num2 < 0) {
		num2 = -num2;
	}

	// Default value, if no common denominator, return 1
	int gcd = 1;

	for (int i = 1; i <= num1 && i <= num2; i++)
	{
		if (num1 % i == 0 && num2 % i == 0)
			gcd = i;
	}
	return gcd;
}
```

Task 4

I opted for two functions, check SSN which basically just verifies the user's input and ssnManOrWoman which checks if the number belongs to a man or a woman. If the second to last number is even it belongs to a woman and if it's odd it belongs to a man.

```
//---------------------------------------------------------------------
//---------------------------------------------------------------------
// File: Task_4.cpp
// Summary: This program checks if a social security number belongs to a
//                  man or a woman
// Version: 1.1
// Owner: Jesper Malmberg
//---------------------------------------------------------------------
// Log: 2021-12-20 Created file
//---------------------------------------------------------------------
// Preprocessor directives
#include <iostream>
#include <cstdlib> // atoi()
#include <cctype> // isdigit()

using namespace std;
// Prototypes
bool checkSsn(char*);
void ssnManOrWoman(char*);

// Constants
const int MAXLEN = 99; // Length of array

int main()
{


	char str[MAXLEN]{ '\0' };
	char answer;			// Continue or stop program variable
	bool ssnOk;

		do {
			system("CLS");
			cout << "Please enter a social security number: ";

			do {
				// Char array for input
				char ssn[MAXLEN] = { '\0' };

				// Read input
				cin.getline(ssn, MAXLEN);

				ssnOk = checkSsn(ssn);

				if (!ssnOk) {
					cout << "Invalid number, please enter the number in
format: YYMMDD-XXXX:";
				}

				if (ssnOk) {
					ssnManOrWoman(ssn);
```

```
                        }

                } while (!ssnOk);

                cout << endl;
                cout << "Want to check another Social Security Number (Y/N)? ";
                cin.get(answer);

                // Clear the input buffer of errors and old data
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
        } while(toupper(answer) == 'Y');

    return 0;
}

//-------------------------------------------------------------------------
// bool checkSsn(char *ssn)
// Checks that an entered ssn number is ok
// Takes an array pointer
// Returns a boolean
//-------------------------------------------------------------------------
bool checkSsn(char *ssn) {
    // Check if entered number is too long
    if (ssn[11]) {
        return false;
    }

    // Check if a hyphen is present and remove it if so
    if (ssn[6] == '-') {
        for (int i = 7; i < 10; i++) {
            ssn[i - 1] = ssn[i];
        }
    }
    // Check that all char are digits
    for (int i = 0; i < 10; i++) {
        if (!isdigit(ssn[i])) {
            return false;
        }
    }
    return true;
}

//-------------------------------------------------------------------------
// void ssnManOrWoman(char* ssn)
// Checks if a SSN belongs to a man or woman
// Takes an array pointer
//-------------------------------------------------------------------------
void ssnManOrWoman(char* ssn) {
    // If even it belongs to a woman
    if (ssn[8] % 2 == 0) {
        cout << "The SSN ";
        for (int i = 0; i < 10; i++) {
            cout << ssn[i];
        }
        cout << " belongs to a woman";
    }
    else {
```

```
			// Else it belongs to a man
			cout << "The SSN ";
			for (int i = 0; i < 10; i++) {
				cout << ssn[i];
			}
			cout << " belongs to a man";
		}
}
```

Task 5

Only one function for this program. Bool isPalindrom basically takes the input char array, writes it backwards in a new array and the original to another removing whitespaces and changes all characters to upper. It then compares char by char if they're the same and returns true if so.

```
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
// File: Task_5.cpp
// Summary: This program checks if a word is a palindrome
// Version: 1.1
// Owner: Jesper Malmberg
//---------------------------------------------------------------------------
// Log: 2021-12-21 Created file
//---------------------------------------------------------------------------
// Preprocessor directives
#include <iostream>


using namespace std;
// Prototypes
bool isPalindrome(char*);

const int MAXLEN = 100; // Max length of the char array

int main()
{
	char answer;

	do {
		system("CLS");			// Clear the console

		char str[MAXLEN]{'\0'};	// Empty the array

		cout << "Enter a word or a phrase: ";

		cin.getline(str, MAXLEN);

		// Check if palindrome and print out as apropriatey
		if (isPalindrome(str)) {
			cout << "It is a palindrome";
		} else {
			cout << "It is not a palidrome";
		}

		cout << endl;
		cout << "Try Again (Y/N)? ";
```

```
			cin.get(answer);

			// Clear the input buffer of errors and old data
			cin.clear();
			cin.ignore(numeric_limits<streamsize>::max(), '\n');
		} while (toupper(answer) == 'Y');

		return 0;
}


//------------------------------------------------------------------------
// bool isPalindrome(char *inpt)
// This function checks whether or not a word or phrase is a palindrome
// Takes a char array as input
// Returns a boolean
//------------------------------------------------------------------------
bool isPalindrome(char *inpt) {

		// Array for backwards and copy of the input
		char backWards[MAXLEN]{'\0'};
		char copy[MAXLEN]{'\0'};
		// Indexes for copying
		int indexOriginal = 0;
		int indexCopy = 0;

		// Copy the input array backwards to backWards[]
		for (int i = strlen(inpt) - 1; i >= 0; i--) {
			if (inpt[i] != ' ') {
				backWards[indexCopy] = toupper(inpt[i]);
				indexCopy++;
			}
		}

		// Copy the input array to copy[]
		for (int i = 0; i < strlen(inpt ); i++) {
			if (inpt[i] != ' ') {
				copy[indexOriginal] = toupper(inpt[i]);
				indexOriginal++;
			}
		}

		// Compare the two arrays char by char to see if they match
		for (int i = 0; i < strlen(copy); i++) {
			if (copy[i] != backWards[i]) {
				return false;
			}
		}
		return true;
}
```

Task 6

I created a structure named Student which has 5 members and one method. There are two main functions, enterData and printData. I basically make the user enter the scores for 3 students which is stored in each structures members. Then just before printing the results the method totalNumericScore

is called which calculates the score and also assign a grade to each student. I had to declare the structure Student before the functions to make the program work.

```
//--------------------------------------------------------------------------
//--------------------------------------------------------------------------
// File: Task_6.cpp
// Summary: This program stores and calculates stutents' average scores
// Version: 1.1
// Owner: Jesper Malmberg
//--------------------------------------------------------------------------
// Log: 2021-12-22 Created file
//--------------------------------------------------------------------------
// Preprocessor directives
#include <iostream>

using namespace std;

// Constants
const int QUIZZES = 2;
const int EXAMS = 2;
const int NUMBEROFSTUDENTS = 3;

// The datastructure Student holds information on a student
struct Student {
	int id;                                          // The id of the Student
	double quizzes[QUIZZES];  // An array with 2 quiz results
	double exams[EXAMS];             // An array with 2 exam results
	double averageScore;             // The average score
	char grade;                              // Final grade

	// Calculates the average or total score as well as the final grade
	// Total score(average score) is calculated as follows:
	// Final exam (exams[1]) = 50%
	// Midterm exam (exams[0]) = 25%
	// Average of the two quizzes = 25%
	// Grade >= 90 = A, 90 >= 80 = B, 80 >= 70 = C, 70 >= 60 = D, <60 = F
	void totalNumericScore() {
		double examWorth = exams[1] * 0.5  + exams[0] * 0.25;
		double quizWorth = (quizzes[0] + quizzes[1]) / 2 * 10 * 0.25;
		averageScore = examWorth + quizWorth;
		if (averageScore >= 90) {
			grade = 'A';
		}
		else if (averageScore < 90 && averageScore >= 80) {
			grade = 'B';
		}
		else if (averageScore < 80 && averageScore >= 70) {
			grade = 'C';
		}
		else if (averageScore < 70 && averageScore >= 60) {
			grade = 'D';
		}
		else if (averageScore < 60) {
			grade = 'F';
		}
	}
};
```

```
// Prototypes
void enterData(Student&);
void printData(Student);

int main()
{
	char answer;

	struct Student students[NUMBEROFSTUDENTS];

	//Student students[NUMBEROFSTUDENTS];

	do {
		system("CLS");			// Clear the console

		// Enter the data tabout each student's quizzes and exams
		for (Student& stu : students) {
			enterData(stu);
		}

		// Print all student records
		for (Student stu : students) {
			stu.totalNumericScore();
			printData(stu);
			cout << endl;
		}

		cout << "Enter more students? (Y/N) ";
		// Clear the input buffer of errors and old data
		std::cin.clear();
		std::cin.ignore(numeric_limits<streamsize>::max(), '\n');
		cin.get(answer);
		// Clear the input buffer of errors and old data
		std::cin.clear();
		std::cin.ignore(numeric_limits<streamsize>::max(), '\n');
	} while (toupper(answer) == 'Y');

	return 0;
}

//-------------------------------------------------------------------------
// void enterData(Student&)
// Enter the record of a student
// Parameter Student is passed by reference to relevant student structure
//-------------------------------------------------------------------------
void enterData(Student& stu) {
	cout << "Enter the student number: ";
	cin >> stu.id;
	cout << "Enter two 10 point quizzes" << endl;
	cin >> stu.quizzes[0] >> stu.quizzes[1];
	cout << "Enter the midterm and final exam grades. These are 100 point tests"
<< endl;
	std::cin >> stu.exams[0] >> stu.exams[1];
}

//-------------------------------------------------------------------------
// void printData(Student)
// Prints the record of all students scores, total score and grade
```

```
//---------------------------------------------------------------------------
void printData(Student stu) {
	cout << "The record for student number : " << stu.id << endl;
	cout << "The quiz grades are : " << stu.quizzes[0] << " " << stu.quizzes[1]
<< endl;
	cout << "The midterm and exam grades are: " << stu.exams[0] << " " <<
stu.exams[1] << endl;
	cout << "The total numeric score is : " << stu.averageScore << endl;
	cout << "and the letter grade assigned is: " << stu.grade << endl;
}
```

Step 4b

Task 1

I chose a max length of the input word to 100 characters as that seems enough. The input word is stored in a char array. I then simply loop over the string backwards and use a char pointer pointing to the backwards array to copy each character and then incrementing the pointer by 1 for each turn in the loop. When loop is finished the pointer point to an 'empty' index so I put the '\0' character there.

```
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
// File: Task_1b.cpp
// Summary: This program takes a word and prints it backwards
// Version: 1.1
// Owner: Jesper Malmberg
//---------------------------------------------------------------------------
// Log: 2021-12-22 Created file
//---------------------------------------------------------------------------
// Preprocessor directives
#include <iostream>

using namespace std;
// Prototypes

const int MAXLEN = 100; // Max length of the char array

int main()
{
	char answer;

	do {
		system("CLS");				// Clear the console

		char str[MAXLEN]{ '\0' }; // Empty the array

		cout << "Enter a text: ";

		// Get the input
		cin.getline(str, MAXLEN);
```

```
            // The word will be stored backards in the backWards array
            char backWards[MAXLEN]{ '\0' };


            char *ptr; // The pointer
            ptr = backWards; // Points to the backwards char array

            int index = 0;

            // Copy the input array backwards to backWards[]
            // Loop starts at the end of the input string and works backwards
            for (int i = strlen(str) - 1; i >= 0; i--) {
                  *ptr = str[i];
                  ptr++; // Increment by one for each loop
            }

            // Pointer incremet is last index of array when above loop finishes so
I add '\0' here.
            *ptr = '\0';

            // Print the backwards text to the terminal
            cout << "The text backwards: ";
            for (char c : backWards) {
                  cout << c;
            }

            cout << endl;
            cout << "Try Again (Y/N)? ";

            cin.get(answer);

            // Clear the input buffer of errors and old data
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
      } while (toupper(answer) == 'Y');

      return 0;
}
```

Task 2

For this task I basically loop over the input string to find the whitespace separating the two names. When the whitespace is found, everything to the right of and including the whitespace is copied to index 1 hence deleting all but the first letter of the first name. I store the new 'end index' in the variable named index. I iterate backwards in the resulting string 'deleting' extra letters with '\0' character until the end is reached and marked by the final '\0'.

```
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
// File: Task_2b.cpp
// Summary: This abbreviates name inputs
// Version: 1.1
// Owner: Jesper Malmberg
//---------------------------------------------------------------------------
// Log: 2021-12-24 Created file
```

```cpp
//---------------------------------------------------------------------------
// Preprocessor directives
#include <iostream>

using namespace std;
// Prototypes

const int MAXLEN = 100; // Max length of the char array

int main()
{
	char answer;

	do {
		system("CLS");				// Clear the console

		char str[MAXLEN]{ '\0' }; // Empty the array

		cout << "Enter a full name: ";

		// Get the input
		cin.getline(str, MAXLEN);

		// Length of original string
		int lenght = strlen(str);

		int index = 0;
		// Find the whitespace and copy everything including whitespace to
index 1
		for (int i = 0; i < strlen(str); i++) {
			if (isspace(str[i])) {
				copy(str + i, str + lenght, str + 1);
				index = strlen(str) - i;

				// Iterate backwards and move end of string to new
position
				for (int j = strlen(str); j > index; j--) {
					str[j] = '\0';
				}
			}
		}

		cout << endl;

		cout << "Treated name: ";
		for (char c : str) {
			cout << c;
		}

		cout << endl;
		cout << "Try Again (Y/N)? ";

		cin.get(answer);

		// Clear the input buffer of errors and old data
		cin.clear();
		cin.ignore(numeric_limits<streamsize>::max(), '\n');
	} while (toupper(answer) == 'Y');
```

```
	return 0;
}
```

Task 3

I chose a max string size of 1000 characters for this task. In the function trimWhitespace I iterate the entire input string and once a whitespace is encountered it enters a while loop. It will copy everything to the right of the analyzed index to the current index (the whitespace) until the index to the right is no longer a whitespace. When that happens the loop breaks. This will possibly result in a string starting and ending with a whitespace, which is trimmed in the following two if statements. The printout prints '*' to indicate the presence of a whitespace for clarity. I had to use "_CRT_SECURE_NO_WARNINGS" in the preprocessor definitions as strcpy generated some warnings in Windows.

```
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
// File: Task_3b.cpp
// Summary: This program removes whitespaces
// Version: 1.1
// Owner: Jesper Malmberg
//---------------------------------------------------------------------------
// Log: 2021-12-24 Created file
//---------------------------------------------------------------------------
// Preprocessor directives
#include <iostream>
#include<cstring>

using namespace std;
// Prototypes
char* trimWhitespace(char* str);
const int MAXLEN = 1000; // Max length of the char array

int main()
{
	char answer;

	do {
		system("CLS");				// Clear the console

		char str[MAXLEN]{ '\0' }; // Empty the array

		cout << "Enter a text: ";
		// Get the input
		cin.getline(str, MAXLEN);
		// Trim input of whitespaces
		trimWhitespace(str);

		cout << endl;

		// Print the trimmed string
		cout << "Cleared text: ";
		for (char c : str) {
			// Print whitespace as a * for clarity
			if (isspace(c)) {
				cout << '*';
			}
```

```
                else {
                        cout << c;
                }
        }

        cout << endl;
        cout << "Try Again (Y/N)? ";

        cin.get(answer);

        // Clear the input buffer of errors and old data
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    } while (toupper(answer) == 'Y');

    return 0;
}

//------------------------------------------------------------------------
// trimwhitespace(*)
// This function takes a pointer to a C string and removes whitespaces
// if there are more than one. This could result in a trimmed string which
// has a whitespace in the beginning and the end. This is removed in the
// second two steps of the function.
//
// Returns a *char pointer to the trimmed string
//------------------------------------------------------------------------
char* trimWhitespace(char* str) {

    char* strPtr = str;             // The string pointer

    // Iterate over the entire string
    for (int i = 0; i < strlen(str); i++) {
        // If the iterated index is a whitespace
        while (isspace(str[i])) {
            // Check if the index next to the iterated is a string, if not
break the loop
            if (!isspace(str[i + 1])) {
                break;
            }
            // Copy substring + 1 index to the iterated index
            strcpy(str + i, strPtr + i + 1);
        }
    }

    // If first index is a whitespace shift array one to the right
    if (isspace(str[0])) {
        strcpy(str, strPtr + 1);
    }

    // If last index is a whitespace replace with null pointer
    if (isspace(str[strlen(str) - 1])) {
        str[strlen(str) - 1] = '\0';
    }
    return str;
}
```

Task 4

Three functions are used in this task. isConsonant which checks for consonants, abbreviate which shortens the word based on the rules and removeConsonant which removes any double consonants. Initially I loop over the input string and remove any vowels from the word. The resulting string of consonants is then processed according to the rules of the assignment. I chose to honor rule 2, no double consonants, above rule 3 which takes the first 3 and last 2 consonants, then checking for the presence of double consonants and removing them resulting in a word shorter than 5. I used strcpy for the copy which required the use of "_CRT_SECURE_NO_WARNINGS" in the preprocessor definitions as strcpy generated some warnings in Windows.

```
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
// File: Task_4b.cpp
// Summary: This program reduces words to acronyms.
// Version: 1.1
// Owner: Jesper Malmberg
//---------------------------------------------------------------------------
// Log: 2022-01-02 Created file
//---------------------------------------------------------------------------
// Preprocessor directives
#include <iostream>
#include<cstring>

using namespace std;
// Prototypes
bool isConsonant(char c);
char* abbreviate(char* str);
char* rmDoubleConsonant(char* str);

const int MAXLEN = 1000; // Max length of the char array
int main()
{
	char answer;

	do {
		system("CLS");                                  // Clear the console

		char str[MAXLEN]{ '\0' };        // Empty the array
		char conString[MAXLEN]{ '\0' }; // Empty the array

		cout << "Enter a text: ";
		// Get the input
		cin.getline(str, MAXLEN);

		// Remove all the vowels in the word
		int index = 0;
		for (char c : str) {
			if (isConsonant(c)) {
				conString[index] = c;
				index++;
			}
		}

		// Abbreviate to military style
		abbreviate(conString);
```

```
			// Print the result
			cout << conString;
			cout << endl;

			cout << endl;
			cout << "Try Again (Y/N)? ";

			cin.get(answer);

			// Clear the input buffer of errors and old data
			cin.clear();
			cin.ignore(numeric_limits<streamsize>::max(), '\n');
		} while (toupper(answer) == 'Y');

		return 0;
}

//------------------------------------------------------------------------
// isConsonant(char)
// Checks if a single character is a consonant and returns true if true
// Returns: Boolean
//------------------------------------------------------------------------
bool isConsonant(char c) {
		char c1 = toupper(c);
		if (c1 == 'A' || c1 == 'O' || c1 == 'U' || c1 == 'E' || c1 == 'I' || c1 ==
'Y') {
			return false;
		}
		return true;
}

//------------------------------------------------------------------------
// abbreviate(char*)
// Abbreviates a consonant string according to the following rules:
// 1. Doubble consonants are treated as 1
// 2. Words with more than 5 consonants are abbreviated keeping the first
// three and last two letters
// 3. Words with 5 or less consonants keep them all unless there are double
//
// Returns: char*
//------------------------------------------------------------------------
char* abbreviate(char* str) {

		// 5 or less letter words
		if (strlen(str) <=5) {
			rmDoubleConsonant(str);
			return str;
		}
		// More than 5 letters
		else {
			strcpy(str + 3, str + strlen(str) - 2);
			rmDoubleConsonant(str);
			return str;
		}
}

//------------------------------------------------------------------------
```

```
// rmDoubleConsonant(char*)
// Checks for more than 1 consonant together and reduces so that there is
// only one. I.e no double consonants.
//
// Returns: char*
//-----------------------------------------------------------------------
char* rmDoubleConsonant(char* str) {

	for (int i = 0; i < strlen(str) - 1; i++) {
		while (str[i] == str[i + 1]) {
			strcpy(str + i, str + i + 1);
		}
	}
	return str;
}
```