TASK 1

StringHolder.h

```
#ifndef STRINGHOLDER_H
#define STRINGHOLDER_H
#include <string>

class StringHolder {
protected:
    std::string m_data;
public:

    StringHolder(std::string str) {
        m_data = str;
    }

    std::string getData() {
        return m_data;
    }

};

#endif
```

StringAnalyzer.h

```
#ifndef STRINGANALYZER_H
#define STRINGANALYZER_H
#include <string>

bool isVowel(char ch);
bool isAlpha(char ch);
bool isUpper(char ch);
bool isLower(char ch);

class StringAnalyzer : public StringHolder {
public:

    StringAnalyzer(std::string str) : StringHolder(str) {}

    int countVowels() {
        int count = 0;
        for (int i = 0; i < m_data.length(); i++) {
            if (isVowel(m_data[i])) count++;
        }
```

```
        return count;
    }

    int countConsonants() {
        int count = 0;
        for (int i = 0; i < m_data.length(); i++) {
            if (isAlpha(m_data[i]) && !isVowel(m_data[i])) count++;
        }
        return count;
    }

    int countUpper() {
        int count = 0;
        for (int i = 0; i < m_data.length(); i++) {
            if (isUpper(m_data[i])) count++;
        }
        return count;
    }

    int countLower() {
        int count = 0;
        for (int i = 0; i < m_data.length(); i++) {
            if (isLower(m_data[i])) count++;
        }
        return count;
    }

};

bool isVowel(char ch) {
    if (ch >= 'A' && ch <= 'Z') ch += 32;
    return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u';
}

bool isAlpha(char ch) {
    return (ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z');
}

bool isUpper(char ch) {
    return ch >= 'A' && ch <= 'Z';
}

bool isLower(char ch) {
    return ch >= 'a' && ch <= 'z';
}

#endif
```

main.cpp

```
#include <iostream>
#include "StringHolder.h"
#include "StringAnalyzer.h"


int main() {

    StringAnalyzer analyzer("ParaLeLepipeD");
    std::cout << "String: " << analyzer.getData() << std::endl;
    std::cout << "Vowels: " << analyzer.countVowels() << std::endl;
    std::cout << "Consonants: " << analyzer.countConsonants() << std::endl;
    std::cout << "Uppercase: " << analyzer.countUpper() << std::endl;
    std::cout << "Lowercase: " << analyzer.countLower() << std::endl;

    return 0;
}
```

TASK 2

CharArray.h

```
#ifndef CHARARRAY_H
#define CHARARRAY_H
#include <cstring>

class CharArray {
protected:
    char* m_arr;
    int m_size;
public:

    CharArray(const char* input) {
        m_size = 0;
        while (input[m_size] != '\0') m_size++;
        m_arr = new char[m_size];
        for (int i = 0; i < m_size; i++) {
            m_arr[i] = input[i];
        }
    }

    ~CharArray() {
```

```
        delete[] m_arr;
    }

    void print() {
        for (int i = 0; i < m_size; i++) {
            std::cout << m_arr[i];
        }
        std::cout << std::endl;
    }

    int getSize() {
        return m_size;
    }

    char getCharAt(int index) {
        if (index >= 0 && index < m_size)
            return m_arr[index];
        return '\0';
    }

};

#endif
```

CharEditor.h

```
#ifndef CHAREDITOR_H
#define CHAREDITOR_H
#include <cstring>

class CharEditor : public CharArray {
public:

    CharEditor(const char* input) : CharArray(input) {}

    void removeChar(char target) {
        int newSize = 0;
        for (int i = 0; i < m_size; i++) {
            if (m_arr[i] != target) newSize++;
        }

        char* newArr = new char[newSize];
        int j = 0;
        for (int i = 0; i < m_size; i++) {
            if (m_arr[i] != target) newArr[j++] = m_arr[i];
        }
```

```
        delete[] m_arr;
        m_arr = newArr;
        m_size = newSize;
    }

};

#endif
```

main.cpp

```
#include <iostream>
#include "CharArray.h"
#include "CharEditor.h"


int main() {

    CharEditor editor("abracadabra");
    std::cout << "Before: ";
    editor.print();
    editor.removeChar('a');
    std::cout << "After: ";
    editor.print();

    return 0;
}
```