Name: Samyam Budhathoki

Reg No: 22BCE3908

Compiler Lab Assessment 1

String Manipulation:

Code:

```cpp
#include <iostream>
#include <cctype>
#include <cstring>

int string_length(const char* str) {
    int length = 0;
    while (str[length] != '\0') {
        length++;
    }
    return length;
}

void string_copy(char* dest, const char* src) {
    while (*src != '\0') {
        *dest = *src;
        dest++;
        src++;
    }
    *dest = '\0';
}

void string_uppercase(char* str) {
    while (*str != '\0') {
        *str = toupper(*str);
        str++;
    }
}

void string_concatenate(char* dest, const char* src) {
    while (*dest != '\0') {
        dest++;
    }
    while (*src != '\0') {
        *dest = *src;
        dest++;
        src++;
    }
    *dest = '\0';
}
```

```
}

int main() {
    char str1[100], str2[100], str3[100];
    std::cout << "Enter the first string: ";
    std::cin.getline(str1, 100);
    std::cout << "Enter the second string: ";
    std::cin.getline(str2, 100);

    std::cout << "Length of '" << str1 << "': " << string_length(str1) << std::endl;

    string_copy(str3, str1);
    std::cout << "Copied string: " << str3 << std::endl;

    string_uppercase(str1);
    std::cout << "Uppercase string: " << str1 << std::endl;

    string_concatenate(str1, str2);
    std::cout << "Concatenated string: " << str1 << std::endl;

    return 0;
}
```

Result:

```
Enter the first string: Samyam
Enter the second string: Budhathoki
Length of 'Samyam': 6
Copied string: Samyam
Uppercase string: SAMYAM
Concatenated string: SAMYAMBudhathoki
```

Token Specification:

```cpp
#include <iostream>
#include <string>
#include <regex>

using namespace std;

int main() {
    string expression;
    cout << "Enter an arithmetic expression: ";
    getline(cin, expression);

    regex token_regex("(\\d+|[+\\-*/%=])");

    // Remove spaces from the expression
    expression.erase(remove(expression.begin(), expression.end(), ' '), expression.end());

    for (sregex_iterator it(expression.begin(), expression.end(), token_regex), end; it != end; ++it) {
        string token = it->str();
        if (isdigit(token[0])) {
            cout << "NUMBER: " << token << endl;
        } else if (token == "=") {
            cout << "EQUALS: " << token << endl;
        } else {
            cout << "OPERATOR: " << token << endl;
        }
    }

    return 0;
}
```

Result:

```
Enter an arithmetic expression: 6 + 90
NUMBER: 6
OPERATOR: +
NUMBER: 90
```

Token Count:

```cpp
#include <iostream>
#include <string>
#include <regex>

using namespace std;

int main() {
    string expression;
    cout << "Enter an arithmetic expression: ";
    getline(cin, expression);

    regex variable_regex("[a-zA-Z]+");
    regex constant_regex("\\d+");
    regex operator_regex("[+\\-*/%=]");

    int variable_count = 0;
    int constant_count = 0;
    int operator_count = 0;


    expression.erase(remove(expression.begin(), expression.end(), ' '), expression.end());

    sregex_iterator it(expression.begin(), expression.end(), variable_regex);
    sregex_iterator end;
    while (it != end) {
        ++variable_count;
        ++it;
    }

    it = sregex_iterator(expression.begin(), expression.end(), constant_regex);
    while (it != end) {
        ++constant_count;
        ++it;
    }

    it = sregex_iterator(expression.begin(), expression.end(), operator_regex);
    while (it != end) {
        ++operator_count;
        ++it;
    }

    cout << "Variable count: " << variable_count << endl;
    cout << "Constant count: " << constant_count << endl;
    cout << "Operator count: " << operator_count << endl;

    return 0;
}
```

Result:

```
Enter an arithmetic expression: y + 60/10
Variable count: 1
Constant count: 2
Operator count: 2
```