Name: Eyasir Ahamed
Exam Roll: 413
Registration No: 202004017
Class Roll: 15

---

1. Design and implement a lexical analyzer in C for a simple calculator that performs addition, subtraction, multiplication, and division. The calculator should handle single-digit and multi-digit integer operands, and the four basic arithmetic operators (+, -, *, /). The lexical analyzer should identify and categorize the following tokens: Integers: Sequences of one or more digits.
Operators: The characters +, -, *, /.
Whitespace: Spaces, tabs, and newlines should be ignored.
Your program should take an arithmetic expression as a string input (e.g., "12 + 3 * 5 - 8 / 2") and output a sequence of recognized tokens, indicating their type and value (if applicable).

Code:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    string s;
    cout << "Math Expression: ";
    getline(cin, s);
    cout << s << endl;
    cout << endl;
    cout<<"Identified token with there type and value"<<endl;
    for (int i = 0; i < s.size(); i++) {
        char c = s[i];
        if (isdigit(c)) {
            string n;
            while (i < s.size() and isdigit(s[i])) {
                n += s[i];
                i++;
            }
            i--;
            cout << "Integer :" << n << endl;
        } else if (c == '+') {
            cout << "Plus :" << c << endl;
        } else if (c == '-') {
            cout << "Minus :" << c << endl;
        } else if (c == '*') {
            cout << "Multiplication :" << c << endl;
        } else if (c == '/') {
            cout << "Division :" << c << endl;
        } else {
            continue;
        }
    }
    return 0;
}
```
Input:
```
12+3*5-8/2
```
Output:
```
Math Expression: 12+3*5-8/2

Identified token with there type and value
Integer :12
Plus :+
Integer :3
Multiplication :*
Integer :5
Minus :-
Integer :8
Division :/
Integer :2
```

2. Write a C program for NFA and DFA and the expression is: (a/b)*abb.

Code:

```
#include <bits/stdc++.h>
using namespace std;

// (a/b)*abb

bool DFA(string s) {
    int state = 0;
    for (auto c : s) {
        switch (state) {
        case 0:
            if (c == 'a') {
                state = 1;
            } else {
                state = 0;
            }
        case 1:
            if (c == 'a') {
                state = 1;
            } else {
                state = 2;
            }
        case 2:
            if (c == 'b') {
                state = 3;
            } else {
                if (c == 'a') {
                    state = 1;
                } else {
                    state = 0;
                }
            }
        case 3:
            if (c == 'b') {
                state = 3;
            } else {
                if (c == 'a') {
                    state = 1;
                } else {
                    state = 0;
                }
            }

        }

    }
    return state == 3;
}

bool NFA(string s) {
    if (s.size() >= 3 and s.substr(s.size() - 3) == "abb") {
        return true;
    }
    return false;
}

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    string s; cin >> s;
    cout << "Given string: " << s << endl;
    if (DFA(s)) {
        cout << "DFA: " << "Accepted" << endl;
    } else {
        cout << "DFA: " << "Rejected" << endl;
    }
    if (NFA(s)) {
        cout << "NFA: " << "Accepted" << endl;
    } else {
        cout << "NFA: " << "Rejected" << endl;
    }
    return 0;
}
```

Input1:

```
abbbbabb
```

Output1:

```
Given string: abbbbabb
DFA: Accepted
NFA: Accepted
```

Input2:

```
abba
```

Output1:

```
Given string: abba
DFA: Rejected
NFA: Rejected
```