C Programm:

A. Check for balance paranthesis in arithmetic expression and tokenize it.

Code:

```
#include <iostream>
#include <stack>
#include <string>

using namespace std;

bool isMatchingPair(char opening, char closing)
{
    return (opening == '(' && closing == ')') || (opening == '{' && closing == '}') || (opening == '[' && closing == ']');
}

bool isBalanced(const string expr)
{
    stack<char> s;

    for (char ch : expr)
    {
        if (ch == '(' || ch == '{' || ch == '[')
        {
            s.push(ch);
        }
        else if (ch == ')' || ch == '}' || ch == ']')
        {
            if (s.empty() || !isMatchingPair(s.top(), ch))
            {
                return false;
            }
            s.pop();
        }
    }
    return s.empty();
}

void tokenize(const string expr)
{
    cout << "Tokens: ";
    for (char ch : expr)
    {
        if (ch != ' ')
        {
            cout << ch << " ";
        }
```

```
    }
    cout << endl;
}

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

    tokenize(expr);

    if (isBalanced(expr))
    {
        cout << "Balanced Parentheses." << endl;
    }
    else
    {
        cout << "Unbalanced Parentheses." << endl;
    }

    return 0;
}
```

Output:

```
PS C:\Users\sdvs1\Desktop\College\Sem - 6\COMPILER CON
Enter an expression: a+b*(c+(d))
Tokens: a + b * ( c + ( d ) )
Balanced Parentheses.
```

Lex:

A. Check for balance paranthesis in arithmetic expression and tokenize it.

Code:

```
%{
#include <stdio.h>
%}

%%
"<"[^>]+">"   { printf("HTML Tag: %s\n", yytext); }
.|\n        ;

%%

int main() {
    yylex();
    return 0;
}

int yywrap() {
    return 1;
}
```

Output:

```
PS C:\Users\sdvs1\Desktop\College\Sem - 6\COMPILER CONSTRUCTION\Internal_Practical> .\a.exe
my<html>tag
HTML Tag: <html>
my<html>tag is <A> 1<span>0 and -<div>+
HTML Tag: <html>
HTML Tag: <A>
HTML Tag: <span>
HTML Tag: <div>
```