# DS TUT 5

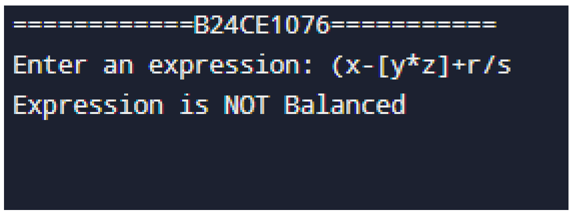
**Name:Varad Tanawade**

**CODE 1:**

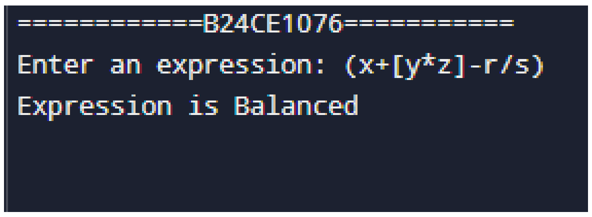
```
/*Parenthesis Checker: Write a program using a stack for push, pop,
peek, and isEmpty operations. Write isBalanced() Function that
Iterates through the input expression, Pushes opening brackets onto
the stack. For closing brackets, it checks the top of the stack for a
matching opening bracket. Ensures that all opening brackets are
matched by the end of the traversal. Main Function: Accepts a string
expression from the user. Uses isBalanced() to determine if the
parentheses in the expression are balanced. */
#include <iostream>
using namespace std;
class Checker {
 char stack[100];
 int top;
 int N;
public:
 Checker() {
 top = -1;
 N = 100;
 }
 int isEmpty() {
 if (top == -1)
 return 1;
 else
 return 0;
 }
 int isFull() {
 if (top == N - 1)
 return 1;
 else
 return 0;
 }
 void push(char ch) {
 if (isFull())
 cout << "Stack overflow\n";
 else {
 top++;
 stack[top] = ch;
 }
 }
 char pop() {
 if (isEmpty()) {
```

```
 cout << "Stack underflow\n";
 return '\0';
 } else {
 char temp = stack[top];
 top--;
 return temp;
 }
 }
 char peek() {
 if (isEmpty())
 return '\0';
 else
 return stack[top];
 }
 int isBalanced(string exp) {
 for (int i = 0; i < exp.length(); i++) {
 char ch = exp[i];
 if (ch == '(' || ch == '{' || ch == '[')
 push(ch);
 else if (ch == ')' || ch == '}' || ch == ']') {
 if (isEmpty())
 return 0;
 char t = peek();
 if ((ch == ')' && t == '(') || (ch == '}' && t == '{') || (ch == ']'
&& t == '['))
 pop();
 else
 return 0;
 }
 }
 if (isEmpty())
 return 1;
 else
 return 0;
 }
};
int main() {
 cout<<"===========B24CE1076===========\n";
 Checker c;
 string expr;
 cout << "Enter an expression: ";
 cin >> expr;
 if (c.isBalanced(expr))
 cout << "Expression is Balanced" << endl;
 else
 cout << "Expression is NOT Balanced" << endl;
 return 0;
}
```

**OUTPUT 1:**



```
============B24CE1076===========
Enter an expression: (x-[y*z]+r/s
Expression is NOT Balanced
```

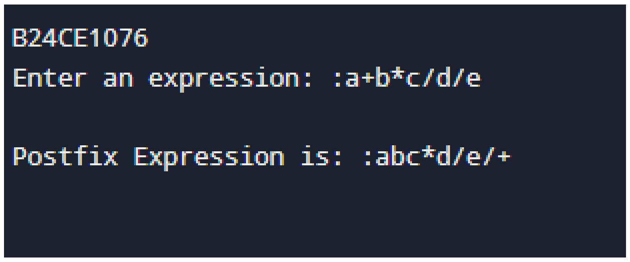


```
============B24CE1076===========
Enter an expression: (x+[y*z]-r/s)
Expression is Balanced
```

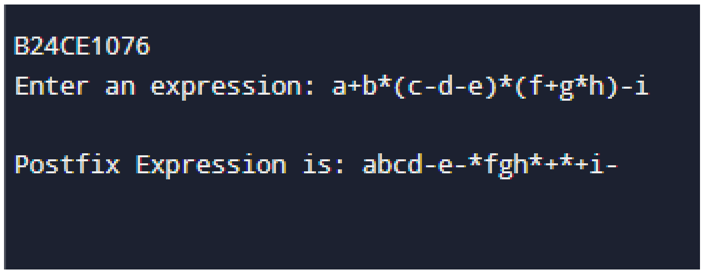
**CODE 2:**

```
#include <iostream>
#include <cstring>
#include <string>
#define N 50
using namespace std;

class Stack {
public:
    char arr[N];
    int top;

    Stack() {
        top = -1;
    }

    void push(char c) {
        if (top == N - 1)
            cout << "Stack overflow\n";
        else
            arr[++top] = c;
    }

    char pop() {
        if (top == -1)
            return '\0';
        return arr[top--];
    }

    char peek() {
        if (top == -1)
            return '\0';
        return arr[top];
    }

    int precedence(char opr) {
        if (opr == '*' || opr == '/')
            return 2;
        if (opr == '+' || opr == '-')
            return 1;
        if (opr == '(')
            return 0;
        return -1;
    }
    string InfixToPostfixConversion(string ex) {
        string op_exp = "";
        char ch, ch1;

        for (int i = 0; i < ex.length(); i++) {
            if (ex[i] == '+' || ex[i] == '-' || ex[i] == '*' || ex[i]
== '/') {
                while (top != -1 && precedence(peek()) >=
precedence(ex[i])) {
                    op_exp += pop();
                }
                push(ex[i]);
            } else if (ex[i] == '(') {
                push(ex[i]);
            } else if (ex[i] == ')') {
                while (top != -1 && peek() != '(') {
                    op_exp += pop();
                }
                pop();
            } else {
                op_exp += ex[i];
            }
        }

        while (top != -1) {
            op_exp += pop();
        }

        return op_exp;
    }
};

int main() {
    Stack s;
    string ex;
    cout<<"\nB24CE1076";
    cout << "\nEnter an expression: ";
    cin >> ex;
    string op_exp = s.InfixToPostfixConversion(ex);
    cout << "\nPostfix Expression is: " << op_exp << endl;
    return 0;
}
```

**OUTPUT 2:**



```
B24CE1076
Enter an expression: :a+b*c/d/e

Postfix Expression is: :abc*d/e/+
```



```
B24CE1076
Enter an expression: a+b*(c-d-e)*(f+g*h)-i

Postfix Expression is: abcd-e-*fgh*+*+i-
```