

# DATA LAB#5

f24-0767

## Task 1:

```
#include <iostream>
#include <string>
using namespace std;

class Stack {
    char arr[1000];
    int top;
public:
    Stack()
    {
        top = -1;
    }
    void push(char c)
    {
        arr[++top] = c;
    }
    char pop()
    {
        return arr[top--];
    }
    bool empty()
    {
        return top == -1;
    }
    char tope()
    {
        return arr[top];
    }
};

bool ismatched(char open, char close)
{
    if (open == '(' && close == ')')
    {
        return true;
    }
    else if (open == '{' && close == '}')
    {
        return true;
    }
    else if (open == '[' && close == ']')
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

```

    }
}

bool validate(string& code)
{
    Stack st;
    for (int i = 0; i < code.length(); i++)
    {
        char c = code[i];
        if (c == '(' || c == '{' || c == '[')
        {
            st.push(c);
        }
        else if (c == ')' || c == '}' || c == ']')
        {
            if (st.empty())
            {
                return false;
            }
            if (!ismatched(st.pop(), c))
            {
                return false;
            }
        }
    }
    return st.empty();
}

int main() {
    string code, line;
    cout << "Enter C++ code:\n";

    while (true) {
        getline(cin, line);
        if (line == "") break;
        code += line + "\n";
    }

    if (validate(code))
    {
        cout << "Valid\n";
    }
    else
    {
        cout << "Invalid\n";
    }

    system("pause");
}

```

```
C:\Users\Laptop Zone\source' x + v
Enter C++ code:
#include<iostream>
using namespace std;
int main()
{
system("pause");
}

Valid
Press any key to continue . . . |
```

## Task 2:

```
#include <iostream>
#include <string>
using namespace std;

class Stack {
    char arr[100];
    int top;
public:
    Stack() { top = -1; }
    void push(char c)
    {
        arr[++top] = c;
    }
    char pop()
    {
        return arr[top--];
    }
    char tope()
    {
        return arr[top];
    }
    bool empty()
    {
        return top == -1;
    }
};

class intstack {
    int arr[100];
    int top;
public:
    intstack()
    {
```

```

        top = -1;
    }
    void push(int x)
    {
        arr[++top] = x;
    }
    int pop()
    {
        return arr[top--];
    }
    bool empty()
    {
        return top == -1;
    }
};

int precedence(char op)
{
    if (op == '+' || op == '-') return 1;
    if (op == '*' || op == '/') return 2;
    return 0;
}

bool isoperator(char c)
{
    return (c == '+' || c == '-' || c == '*' || c == '/');
}

string infixtopostfix(string infix)
{
    Stack st;
    string postfix = "";

    for (int i = 0; i < infix.length(); i++)
    {
        char c = infix[i];

        if (c == ' ')
        {
            // skip spaces
        }
        else if (isdigit(c)) {
            string num = "";
            while (i < infix.length() && isdigit(infix[i])) {
                num += infix[i];
                i++;
            }
            i--;
            postfix += num;
            postfix += ' ';
        }
        else if (c == '(') {
            st.push(c);
        }
        else if (c == ')') {
            while (!st.empty() && st.tope() != '(') {
                postfix += st.pop();
                postfix += ' ';
            }
        }
    }
}

```

```

        if (st.empty()) return "Invalid";
        st.pop();
    }
    else if (isoperator(c))
    {
        while (!st.empty())
        {
            char topop = st.tope();
            if (precedence(topop) >= precedence(c))
            {
                postfix += st.pop();
                postfix += ' ';
            }
            else {
                break;
            }
        }
        st.push(c);
    }
    else {
        return "Invalid";
    }
}

while (!st.empty()) {
    if (st.tope() == '(')
        return "Invalid";

    postfix += st.pop();
    postfix += ' ';
}

return postfix;
}

```

```

bool evaluate(string postfix, int& answer) {
    intstack st;

    for (int i = 0; i < postfix.length(); i++) {
        char c = postfix[i];

        if (c == ' ') {
        }
        else if (isdigit(c)) {
            int num = 0;
            while (i < postfix.length() && isdigit(postfix[i])) {
                num = num * 10 + (postfix[i] - '0');
                i++;
            }
            i--;
            st.push(num);
        }
        else if (isoperator(c)) {
            if (st.empty())
                return false;
            int b = st.pop();
            if (st.empty())
                return false;
            int a = st.pop();

```

```

        int result = 0;
        if (c == '+')
            result = a + b;
        else if (c == '-')
            result = a - b;
        else if (c == '*')
            result = a * b;
        else if (c == '/')
        {
            if (b == 0)
                return false;
            result = a / b;
        }
        st.push(result);
    }
    else {
        return false;
    }
}

if (st.empty()) return false;
answer = st.pop();
if (!lst.empty()) return false;
return true;
}

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

    string postfix = infixtopostfix(infix);

    if (postfix == "Invalid")
    {
        cout << "Invalid expression\n";
    }
    else
    {
        cout << "Postfix: " << postfix << endl;

        int result;
        if (evaluate(postfix, result))
        {
            cout << "Result: " << result << endl;
        }
        else
        {
            cout << "Invalid expression\n";
        }
    }

    system("pause");
}

```

C:\Users\Laptop Zone\source' × + ▾

Enter infix expression:  $7/5 + (4 - (2) * 3$

Invalid expression

Press any key to continue . . .

```
C:\Users\Laptop Zone\source' x + v
Enter infix expression: 10+3*5/(16-4)
Postfix: 10 3 5 * 16 4 - / +
Result: 11
Press any key to continue . . . |
```

### Task 3:

```
#include <iostream>
#include <string>
using namespace std;

class Stack {
    char arr[100];
    int top;
public:
    Stack()
    {
        top = -1;
    }
    void push(char c)
    {
        arr[++top] = c;
    }
    char pop()
    {
        return arr[top--];
    }
    char tope()
    {
        return arr[top];
    }
}
```

```

        bool empty()
        {
            return top == -1;
        }
};

string remover(string s)
{
    Stack st;

    for (int i = 0; i < s.length(); i++)
    {
        char c = s[i];

        if (!st.empty() && st.tope() == c)
        {
            st.pop();
        }
        else
        {
            st.push(c);
        }
    }

    string result = "";

    while (!st.empty())
    {
        result = st.pop() + result;
    }

    return result;
}

int main()
{
    string s;
    cout << "Enter string: ";
    cin >> s;

    string ans = remover(s);

    cout << "Result: " << ans << endl;

    system("pause");
}

```

C:\Users\Laptop Zone\source' x

+

▼

Enter string: acddac

Result: acac

Press any key to continue . . .

C:\Users\Laptop Zone\source' x

+

▼

Enter string: abbaca

Result: ca

Press any key to continue . . .