**AQ1**

#include <iostream>

using namespace std;

const int MAX\_SIZE = 100;

int stack[MAX\_SIZE];

int top = -1;

void push(int value)

{

    if (top >= MAX\_SIZE - 1)

    {

        cout << "Stack Overflow! Cannot push " << value << endl;

    }

    else

    {

        top++;

        stack[top] = value;

        cout << "Pushed " << value << " onto the stack" << endl;

    }

}

int pop()

{

    if (top < 0)

    {

        cout << "Stack Underflow! Stack is empty" << endl;

        return -1;

    }

    else

    {

        int value = stack[top];

        top--;

        cout << "Popped " << value << " from the stack" << endl;

        return value;

    }

}

bool isEmpty()

{

    return (top < 0);

}

bool isFull()

{

    return (top >= MAX\_SIZE - 1);

}

void display()

{

    if (isEmpty())

    {

        cout << "Stack is empty" << endl;

    }

    else

    {

        cout << "Stack elements: ";

        for (int i = top; i >= 0; i--)

        {

            cout << stack[i] << " ";

        }

        cout << endl;

    }

}

int peek()

{

    if (isEmpty())

    {

        cout << "Stack is empty" << endl;

        return -1;

    }

    else

    {

        cout << "Top element: " << stack[top] << endl;

        return stack[top];

    }

}

int main()

{

    int choice, value;

    while (true)

    {

        cout << "\tStack Operations Menu" << endl;

        cout << "1. Push" << endl;

        cout << "2. Pop" << endl;

        cout << "3. Check if Empty" << endl;

        cout << "4. Check if Full" << endl;

        cout << "5. Display Stack" << endl;

        cout << "6. Peek (Top Element)" << endl;

        cout << "7. Exit" << endl;

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice)

        {

        case 1:

            cout << "Enter value to push: ";

            cin >> value;

            push(value);

            break;

        case 2:

            pop();

            break;

        case 3:

            if (isEmpty())

            {

                cout << "Stack is empty" << endl;

            }

            else

            {

                cout << "Stack is not empty" << endl;

            }

            break;

        case 4:

            if (isFull())

            {

                cout << "Stack is full" << endl;

            }

            else

            {

                cout << "Stack is not full" << endl;

            }

            break;

        case 5:

            display();

            break;

        case 6:

            peek();

            break;

        case 7:

            cout << "Exiting program..." << endl;

            return 0;

        default:

            cout << "Invalid choice! Please try again." << endl;

        }

    }

    return 0;

}

**Output:**

**AQ2**

#include <iostream>

#include <stack>

using namespace std;

int main() {

    string s = "DataStructure";

    stack<char> st;

    for (char c : s)

        st.push(c);

    string res;

    while (!st.empty()) {

        res += st.top();

        st.pop();

    }

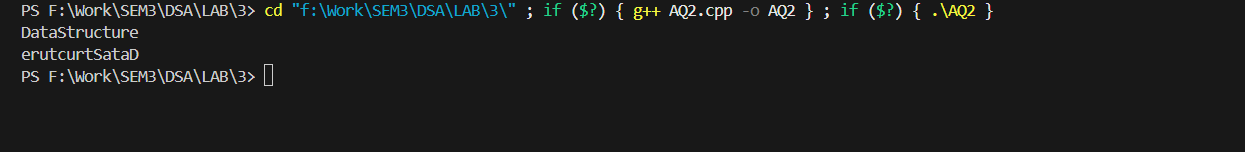
    cout << s << endl;

    cout << res << endl;

    return 0;

}

**Output:**

****

**AQ3**

#include <iostream>

#include <stack>

using namespace std;

int main() {

    string s;

    cout << "Enter expression: ";

    cin >> s;

    stack<char> st;

    bool ch = true;

    for (char c : s)

    {

        if (c == '(' || c == '{' || c == '[')

        {

            st.push(c);

        }

        else if (c == ')' || c == '}' || c == ']')

        {

            if (st.empty())

            {

                ch = false;

                break;

            }

            char t = st.top();

            st.pop();

            if ((c == ')' && t != '(') || (c == '}' && t != '{') || (c == ']' && t != '['))

            {

                ch = false;

                break;

            }

        }

    }

    if (!st.empty()) ch = false;

    if (ch)

    {

        cout << "Balanced";

    }

    else

    {

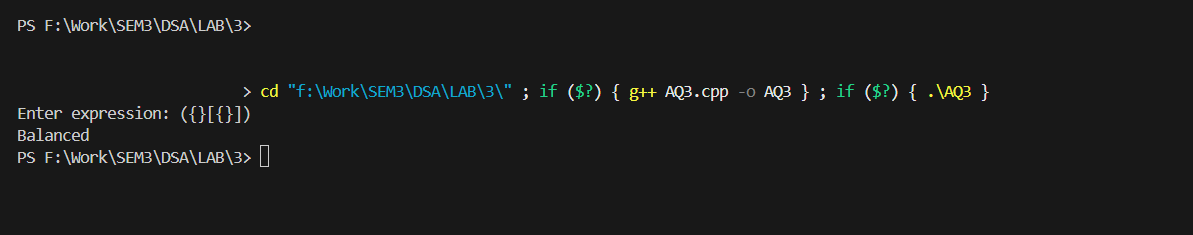
        cout << "Not Balanced";

    }

    return 0;

}

**Output:**

****

**AQ4**

#include <iostream>

#include <stack>

#include <string>

using namespace std;

int main() {

    string s;

    cout << "Enter infix expression: ";

    cin >> s;

    stack<char> st;

    string res;

    for (int i = 0; i < s.length(); i++) {

        char c = s[i];

        if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9'))

            res += c;

        else if (c == '(')

            st.push('(');

        else if (c == ')') {

            while (st.top() != '(') {

                res += st.top();

                st.pop();

            }

            st.pop();

        }

        else {

            int prec\_c;

            if (c == '^')

                prec\_c = 3;

            else if (c == '/' || c == '\*')

                prec\_c = 2;

            else if (c == '+' || c == '-')

                prec\_c = 1;

            else

                prec\_c = -1;

            while (!st.empty()) {

                char top = st.top();

                int prec\_top;

                if (top == '^')

                    prec\_top = 3;

                else if (top == '/' || top == '\*')

                    prec\_top = 2;

                else if (top == '+' || top == '-')

                    prec\_top = 1;

                else

                    prec\_top = -1;

                if (prec\_c <= prec\_top) {

                    res += st.top();

                    st.pop();

                } else {

                    break;

                }

            }

            st.push(c);

        }

    }

    while (!st.empty()) {

        res += st.top();

        st.pop();

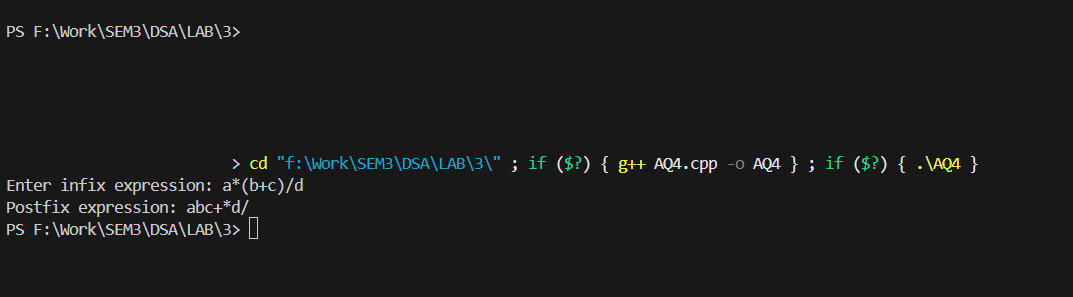
    }

    cout << "Postfix expression: " << res << endl;

    return 0;

}

**Output:**

****

**AQ5**

#include <iostream>

#include <stack>

#include <string>

using namespace std;

int main()

{

    string input;

    cout << "Enter postfix expression: ";

    getline(cin, input);

    stack<int> st;

    string token = "";

    for (int i=0;i<=input.length();i++)

    {

        if (i==input.length() || input[i]==' ')

        {

            if (!token.empty())

            {

                if (isdigit(token[0]) || (token.size()>1 && token[0]=='-'))

                {

                    int num = 0;

                    int sign = 1;

                    int start = 0;

                    if (token[0]=='-')

                    {

                        sign = -1;

                        start = 1;

                    }

                    for (int j=start;j<token.length(); j++)

                    {

                        num=num\*10+(token[j]-'0');

                    }

                    st.push(sign\*num);

                }

                else

                {

                    int val1 = st.top();

                    st.pop();

                    int val2 = st.top();

                    st.pop();

                    if (token=="+")

                    {

                        st.push(val2 + val1);

                    }

                    else if (token=="-")

                    {

                        st.push(val2 - val1);

                    }

                    else if (token=="\*")

                    {

                        st.push(val2 \* val1);

                    }

                    else if (token=="/")

                    {

                        st.push(val2 / val1);

                    }

                }

                token = "";

            }

        }

        else if (input[i]!=' ')

        {

            token+=input[i];

        }

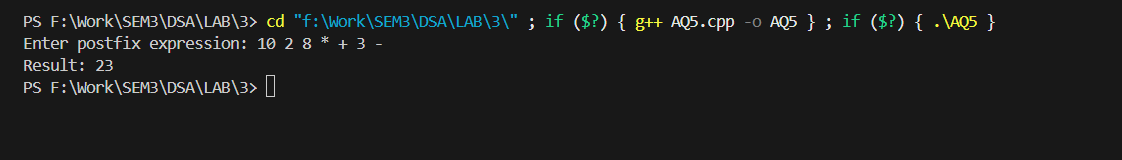
    }

    cout << "Result: " << st.top() << endl;

    return 0;

}

**Output:**

****