Assignment-3

Ayush garg(1024030878)

Q1

main.cpp

```cpp
#include <iostream>
using namespace std;
const int MAX = 100;

class Stack {
    int arr[MAX];
    int top;
public:
    Stack() { top = -1; }

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

    bool isFull() {
        return top == MAX - 1;
    }

    void push(int x) {
        if (isFull()) {
            cout << "Stack Overflow" << endl;
            return;
        }
        arr[++top] = x;
        cout << x << " pushed into stack" << endl;
    }

    void pop() {
        if (isEmpty()) {
            cout << "Stack Underflow" << endl;
            return;
        }
        cout << arr[top--] << "popped from stack" << endl;
    }

    void peek() {
        if (isEmpty()) {
            cout << "Stack is empty" << endl;
            return;
        }
        cout << "Top element: " << arr[top] << endl;
    }

    void display() {
        if (isEmpty()) {
            cout << "Stack is empty!" << endl;
            return;
        }
        cout << "Stack elements: ";
        for (int i = top; i >= 0; i--)
            cout << arr[i] << " ";
        cout << endl;
    }
};

int main() {
```

Output

```
---- Stack Menu ----
1. Push
2. Pop
3. Peek
4. isEmpty
5. isFull
6. Display
0. Exit
Enter choice: 1
Enter value: 20
20 pushed into stack

---- Stack Menu ----
1. Push
2. Pop
3. Peek
4. isEmpty
5. isFull
6. Display
0. Exit
Enter choice: 2
20popped from stack

---- Stack Menu ----
1. Push
2. Pop
3. Peek
4. isEmpty
5. isFull
6. Display
0. Exit
Enter choice: 1
Enter value: 30
30 pushed into stack

---- Stack Menu ----
1. Push
2. Pop
3. Peek
4. isEmpty
5. isFull
6. Display
0. Exit
Enter choice: 3
Top element: 30

---- Stack Menu ----
1. Push
2. Pop
3. Peek
4. isEmpty
5. isFull
```

```cpp
int main() {
    Stack s;
    int choice, value;

    do {
        cout << "\n---- Stack Menu ----\n";
        cout << "1. Push\n2. Pop\n3. Peek\n4. isEmpty\n5. isFull\n6. Display\n0. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            cout << "Enter value: ";
            cin >> value;
            s.push(value);
            break;
        case 2:
            s.pop();
            break;
        case 3:
            s.peek();
            break;
        case 4:
            cout << (s.isEmpty() ? "Stack is empty" : "Stack is not empty") << endl;
            break;
        case 5:
            cout << (s.isFull() ? "Stack is full" : "Stack is not full") << endl;
            break;
        case 6:
            s.display();
            break;
        case 0:
            cout << "Exiting" << endl;
            break;
        default:
            cout << "Invalid choice" << endl;
        }
    } while (choice != 0);

    return 0;
}
```

```
---- Stack Menu ----
1. Push
2. Pop
3. Peek
4. isEmpty
5. isFull
6. Display
0. Exit
Enter choice: 4
Stack is not empty

---- Stack Menu ----
1. Push
2. Pop
3. Peek
4. isEmpty
5. isFull
6. Display
0. Exit
Enter choice: 5
Stack is not full

---- Stack Menu ----
1. Push
2. Pop
3. Peek
4. isEmpty
5. isFull
6. Display
0. Exit
Enter choice: 5
Stack is not full

---- Stack Menu ----
1. Push
2. Pop
3. Peek
4. isEmpty
5. isFull
6. Display
0. Exit
Enter choice: 6
Stack elements: 30

---- Stack Menu ----
1. Push
2. Pop
3. Peek
4. isEmpty
5. isFull
6. Display
0. Exit
Enter choice: 0
Exiting


=== Code Execution Successful ===
```

Q2

main.cpp | Share | Run | Output | Clear

```cpp
#include <iostream>
#include <stack>
using namespace std;

string reverseString(string str) {
    stack<char> s;
    for (char c : str)
        s.push(c);

    string rev = "";
    while (!s.empty()) {
        rev += s.top();
        s.pop();
    }
    return rev;
}

int main() {
    string input;
    cout << "Enter string: ";
    cin >> input;
    cout << "Reversed: " << reverseString(input) << endl;
    return 0;
}
```

```
Enter string: DataStructure
Reversed: erutcurtSataD


=== Code Execution Successful ===
```

Q3

```cpp
#include <iostream>
#include <stack>
using namespace std;

bool isBalanced(string expr) {
    stack<char> s;
    for (char c : expr) {
        if (c == '(' || c == '{' || c == '[') {
            s.push(c);
        } else if (c == ')' || c == '}' || c == ']') {
            if (s.empty()) return false;
            char top = s.top(); s.pop();
            if ((c == ')' && top != '(') ||
                (c == '}' && top != '{') ||
                (c == ']' && top != '['))
                return false;
        }
    }
    return s.empty();
}
int main() {
    string expr;
    cout << "Enter expression: ";
    cin >> expr;
    if (isBalanced(expr))
        cout << "Balanced\n";
    else
        cout << "Not Balanced\n";
```

```
Enter expression: (]
Not Balanced


=== Code Execution Successful ===
```

Q4

main.cpp | Share | Run | Output | Clear

```cpp
#include <iostream>
#include <stack>
using namespace std;

int precedence(char op) {
    if (op == '+' || op == '-') return 1;
    if (op == '*' || op == '/') return 2;
    if (op == '^') return 3;
    return 0;
}

string infixToPostfix(string infix) {
    stack<char> s;
    string postfix = "";

    for (char c : infix) {
        if (isalnum(c)) {
            postfix += c;
        }
        else if (c == '(') {
            s.push(c);
        }
        else if (c == ')') {
            while (!s.empty() && s.top() != '(') {
                postfix += s.top();
                s.pop();
            }
            s.pop();
        }
        else {
            while (!s.empty() && precedence(s.top()) >= precedence(c)) {
                postfix += s.top();
                s.pop();
            }
            s.push(c);
        }
    }
    while (!s.empty()) {
        postfix += s.top();
        s.pop();
    }
    return postfix;
}

int main() {
    string infix;
    cout << "Enter infix expression: ";
    cin >> infix;
    cout << "Postfix: " << infixToPostfix(infix) << endl;
    return 0;
}
```

```
Enter infix expression: A+B*C-D
Postfix: ABC*+D-


=== Code Execution Successful ===
```

Q5

main.cpp

Share Run

```cpp
#include <iostream>
#include <stack>
#include <cmath>
using namespace std;

int evaluatePostfix(string expr) {
    stack<int> s;
    for (char c : expr) {
        if (isdigit(c)) {
            s.push(c - '0');
        } else {
            int val2 = s.top(); s.pop();
            int val1 = s.top(); s.pop();
            switch (c) {
                case '+': s.push(val1 + val2); break;
                case '-': s.push(val1 - val2); break;
                case '*': s.push(val1 * val2); break;
                case '/': s.push(val1 / val2); break;
                case '^': s.push(pow(val1, val2)); break;
            }
        }
    }
    return s.top();
}

int main() {
    string postfix;
    cout << "Enter postfix expression (single-digit operands): ";
    cin >> postfix;
```

Output

Clear

```
Enter postfix expression (single-digit operands): 24+56*-
Result: -24


=== Code Execution Successful ===
```


```cpp
using namespace std;

int evaluatePostfix(string expr) {
    stack<int> s;
    for (char c : expr) {
        if (isdigit(c)) {
            s.push(c - '0');
        } else {
            int val2 = s.top(); s.pop();
            int val1 = s.top(); s.pop();
            switch (c) {
                case '+': s.push(val1 + val2); break;
                case '-': s.push(val1 - val2); break;
                case '*': s.push(val1 * val2); break;
                case '/': s.push(val1 / val2); break;
                case '^': s.push(pow(val1, val2)); break;
            }
        }
    }
    return s.top();
}

int main() {
    string postfix;
    cout << "Enter postfix expression (single-digit operands): ";
    cin >> postfix;
    cout << "Result: " << evaluatePostfix(postfix) << endl;
    return 0;
}
```

Enter postfix expression (single-digit operands): 24+56*-
Result: -24

=== Code Execution Successful ===