

# Assignment-3

## Ayush garg(1024030878)



Q1

```
main.cpp  Run  Output  Clear

1 #include <iostream>
2 using namespace std;
3 const int MAX = 100;
4
5 class Stack {
6     int arr[MAX];
7     int top;
8 public:
9     Stack() { top = -1; }
10
11     bool isEmpty() {
12         return top == -1;
13     }
14
15     bool isFull() {
16         return top == MAX - 1;
17     }
18
19     void push(int x) {
20         if (isFull()) {
21             cout << "Stack Overflow" << endl;
22             return;
23         }
24         arr[++top] = x;
25         cout << x << " pushed into stack" << endl;
26     }
27
28     void pop() {
29         if (isEmpty()) {
30             cout << "Stack Underflow" << endl;
31             return;
32         }
33         cout << arr[top--] << "popped from stack" << endl;
34     }
35
36     void peek() {
37         if (isEmpty()) {
38             cout << "Stack is empty" << endl;
39             return;
40         }
41         cout << "Top element: " << arr[top] << endl;
42     }
43
44     void display() {
45         if (isEmpty()) {
46             cout << "Stack is empty!" << endl;
47             return;
48         }
49         cout << "Stack elements: ";
50         for (int i = top; i >= 0; i--)
51             cout << arr[i] << " ";
52         cout << endl;
53     }
54 };
55
56 int main() {
    ---- 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
```

```

56- int main() {
57-     Stack s;
58-     int choice, value;
59-
60-     do {
61-         cout << "\n---- Stack Menu ----\n";
62-         cout << "1. Push\n2. Pop\n3. Peek\n4. isEmpty\n5. isFull\n6. Display\n0. Exit\n";
63-         cout << "Enter choice: ";
64-         cin >> choice;
65-
66-         switch (choice) {
67-             case 1:
68-                 cout << "Enter value: ";
69-                 cin >> value;
70-                 s.push(value);
71-                 break;
72-             case 2:
73-                 s.pop();
74-                 break;
75-             case 3:
76-                 s.peek();
77-                 break;
78-             case 4:
79-                 cout << (s.isEmpty() ? "Stack is empty" : "Stack is not empty") << endl;
80-                 break;
81-             case 5:
82-                 cout << (s.isFull() ? "Stack is full" : "Stack is not full") << endl;
83-                 s.push(value);
84-                 break;
85-             case 6:
86-                 s.display();
87-                 break;
88-             case 0:
89-                 cout << "Exiting" << endl;
90-                 break;
91-             default:
92-                 cout << "Invalid choice" << endl;
93-         }
94-     } while (choice != 0);
95-     return 0;
96- }

```

```

---- 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: 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
1 #include <iostream>
2 #include <stack>
3 using namespace std;
4
5 string reverseString(string str) {
6     stack<char> s;
7     for (char c : str)
8         s.push(c);
9
10    string rev = "";
11    while (!s.empty()) {
12        rev += s.top();
13        s.pop();
14    }
15    return rev;
16 }
17
18 int main() {
19     string input;
20     cout << "Enter string: ";
21     cin >> input;
22     cout << "Reversed: " << reverseString(input) << endl;
23     return 0;
24 }

```

```

Enter string: DataStructure
Reversed: erutcurtSataD

=== Code Execution Successful ===

```

### Q3

```

1 #include <iostream>
2 #include <stack>
3 using namespace std;
4
5 bool isBalanced(string expr) {
6     stack<char> s;
7     for (char c : expr) {
8         if (c == '(' || c == '[' || c == '{') {
9             s.push(c);
10        } else if (c == ')' || c == ']' || c == '}') {
11            if (s.empty()) return false;
12            char top = s.top(); s.pop();
13            if ((c == ')' && top != '(') ||
14                (c == ']' && top != '[') ||
15                (c == '}' && top != '{'))
16                return false;
17        }
18    }
19    return s.empty();
20 }
21
22 int main() {
23     string expr;
24     cout << "Enter expression: ";
25     cin >> expr;
26     if (isBalanced(expr))
27         cout << "Balanced\n";
28     else
29         cout << "Not Balanced\n";
30 }

```

Enter expression: {}  
Not Balanced

=== Code Execution Successful ===

### Q4

main.cpp

Share

Run

Output

Clear

```

1 #include <iostream>
2 #include <stack>
3 using namespace std;
4
5 int precedence(char op) {
6     if (op == '+' || op == '-') return 1;
7     if (op == '*' || op == '/') return 2;
8     if (op == '^') return 3;
9     return 0;
10 }
11
12 string infixToPostfix(string infix) {
13     stack<char> s;
14     string postfix = "";
15
16     for (char c : infix) {
17         if (isalnum(c)) {
18             postfix += c;
19         }
20         else if (c == '(') {
21             s.push(c);
22         }
23         else if (c == ')') {
24             while (!s.empty() && s.top() != '(') {
25                 postfix += s.top();
26                 s.pop();
27             }
28             s.pop();
29         }
30     }
31
32     while (!s.empty()) {
33         postfix += s.top();
34         s.pop();
35     }
36
37     return postfix;
38 }
39
40 int main() {
41     string infix;
42     cout << "Enter infix expression: ";
43     cin >> infix;
44     cout << "Postfix: " << infixToPostfix(infix) << endl;
45     return 0;
46 }

```

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

=== Code Execution Successful ===

main.cpp

Share

Run

Output

Clear

```

1 #include <iostream>
2 #include <stack>
3 using namespace std;
4
5 int precedence(char op) {
6     if (op == '+' || op == '-') return 1;
7     if (op == '*' || op == '/') return 2;
8     if (op == '^') return 3;
9     return 0;
10 }
11
12 string infixToPostfix(string infix) {
13     stack<char> s;
14     string postfix = "";
15
16     for (char c : infix) {
17         if (isalnum(c)) {
18             postfix += c;
19         }
20         else if (c == '(') {
21             s.push(c);
22         }
23         else if (c == ')') {
24             while (!s.empty() && s.top() != '(') {
25                 postfix += s.top();
26                 s.pop();
27             }
28             s.pop();
29         }
30         else {
31             while (!s.empty() && precedence(s.top()) >= precedence(c)) {
32                 postfix += s.top();
33                 s.pop();
34             }
35             s.push(c);
36         }
37     }
38
39     while (!s.empty()) {
40         postfix += s.top();
41         s.pop();
42     }
43
44     return postfix;
45 }
46
47 int main() {
48     string infix;
49     cout << "Enter infix expression: ";
50     cin >> infix;
51     cout << "Postfix: " << infixToPostfix(infix) << endl;
52     return 0;
53 }

```

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

=== Code Execution Successful ===

## Q5

main.cpp

Share

Run

Clear

```

1 #include <iostream>
2 #include <stack>
3 #include <cmath>
4 using namespace std;
5
6 int evaluatePostfix(string expr) {
7     stack<int> s;
8     for (char c : expr) {
9         if (isdigit(c)) {
10             s.push(c - '0');
11         } else {
12             int val2 = s.top(); s.pop();
13             int val1 = s.top(); s.pop();
14             switch (c) {
15                 case '+': s.push(val1 + val2); break;
16                 case '-': s.push(val1 - val2); break;
17                 case '*': s.push(val1 * val2); break;
18                 case '/': s.push(val1 / val2); break;
19                 case '^': s.push(pow(val1, val2)); break;
20             }
21         }
22     }
23     return s.top();
24 }
25
26 int main() {
27     string postfix;
28     cout << "Enter postfix expression (single-digit operands): ";
29     cin >> postfix;
30
31     using namespace std;
32
33     int evaluatePostfix(string expr) {
34         stack<int> s;
35         for (char c : expr) {
36             if (isdigit(c)) {
37                 s.push(c - '0');
38             } else {
39                 int val2 = s.top(); s.pop();
40                 int val1 = s.top(); s.pop();
41                 switch (c) {
42                     case '+': s.push(val1 + val2); break;
43                     case '-': s.push(val1 - val2); break;
44                     case '*': s.push(val1 * val2); break;
45                     case '/': s.push(val1 / val2); break;
46                     case '^': s.push(pow(val1, val2)); break;
47                 }
48             }
49         }
50         return s.top();
51     }
52
53     int main() {
54         string postfix;
55         cout << "Enter postfix expression (single-digit operands): ";
56         cin >> postfix;
57         cout << "Result: " << evaluatePostfix(postfix) << endl;
58         return 0;
59     }

```

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

Result: -24

=== Code Execution Successful ===