ASSIGMENT 3

QUESTION 1

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

#define MAX 100

class Stack {
    int arr[MAX];
    int top;

public:
    Stack() {
        top = -1;
    }

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

    void pop() {
        if (isEmpty()) {
            cout << "Stack Underflow!\n";
```

```
        return;
    }
    cout << arr[top--] << " popped from stack.\n";
}

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

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

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

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

```
    }
};

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

    do {
        cout << "\n--- Stack Menu ---\n";
        cout << "1. Push\n2. Pop\n3. Peek\n4. Display\n5. Check Empty\n6. Check Full\n7. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            cout << "Enter value to push: ";
            cin >> value;
            s.push(value);
            break;
        case 2:
            s.pop();
            break;
        case 3:
            s.peek();
            break;
        case 4:
            s.display();
            break;
        case 5:
            cout << (s.isEmpty() ? "Stack is Empty.\n" : "Stack is not Empty.\n");
            break;
```

```
        case 6:
            cout << (s.isFull() ? "Stack is Full.\n" : "Stack is not Full.\n");
            break;
        case 7:
            cout << "Exiting program.\n";
            break;
        default:
            cout << "Invalid choice!\n";
        }
    } while (choice != 7);

    return 0;
}
```

QUESTION 2

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

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

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

    return reversed;
}

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

    string output = reverseString(input);
    cout << "Reversed string: " << output << endl;
```

```
    return 0;
}
```

QUESTION 3

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

bool isBalanced(string expr) {
    stack<char> s;
    for (char ch : expr) {
        if (ch == '(' || ch == '{' || ch == '[') {
            s.push(ch);
        }
        else if (ch == ')' || ch == '}' || ch == ']') {
            if (s.empty()) return false;
            char top = s.top();
            s.pop();

            if ((ch == ')' && top != '(') ||
                (ch == '}' && top != '{') ||
                (ch == ']' && top != '['))
                return false;
        }
    }
    return s.empty();
}

int main() {
    string expr;
```

```
    cout << "Enter an expression: ";
    cin >> expr;

    if (isBalanced(expr))
        cout << "Expression has Balanced Parentheses.\n";
    else
        cout << "Expression has Unbalanced Parentheses.\n";

    return 0;
}
```

QUESTION 4

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

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

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

    for (char ch : infix) {
        if (isalnum(ch)) {  // Operand
            postfix += ch;
        }
        else if (ch == '(') {
            s.push(ch);
```

```
        }
        else if (ch == ')') {
            while (!s.empty() && s.top() != '(') {
                postfix += s.top();
                s.pop();
            }
            s.pop(); // Remove '('
        }
        else { // Operator
            while (!s.empty() && precedence(s.top()) >= precedence(ch)) {
                postfix += s.top();
                s.pop();
            }
            s.push(ch);
        }
    }

    while (!s.empty()) {
        postfix += s.top();
        s.pop();
    }

    return postfix;
}

int main() {
    string infix;
    cout << "Enter an infix expression: ";
```

```
    cin >> infix;

    string postfix = infixToPostfix(infix);
    cout << "Postfix expression: " << postfix << endl;

    return 0;
}
```

QUESTION 5

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

int evaluatePostfix(string postfix) {
    stack<int> s;

    for (char ch : postfix) {
        if (isdigit(ch)) {
            s.push(ch - '0');  // Convert char to int
        }
        else {
            int val2 = s.top(); s.pop();
            int val1 = s.top(); s.pop();

            switch (ch) {
                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 a postfix expression (use digits only): ";
    cin >> postfix;

    cout << "Result: " << evaluatePostfix(postfix) << endl;

    return 0;
}
```