Name – Rashi Batch – 2C25

Roll number: 1024030373

Ques 1. Develop a menu driven program demonstrating the following operations on a Stack using array:

1. push(), (ii) pop(), (iii) isEmpty(), (iv) isFull(), (v) display(), and (vi) peek().

Code: #include<iostream>

using namespace std;

const int MAX = 5;

class stack{

private:

    int arr[MAX];

    int top;

public:

    stack(){

        top = -1;

    }

    bool isEmpty(){

        return (top == -1);

    }

    bool isFull(){

        return (top == MAX-1);

    }

    void push(int a){

        if (isFull()) {

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

        } else {

            arr[++top] = a;

            cout << a << " pushed into stack." << endl;

        }

    }

    void pop(){

        if (isEmpty()){

            cout<<"Stack Underflow! Cannot pop"<<endl;

        }

        else{

            cout << arr[top--] << " popped from stack." << endl;

        }

    }

    void peek(){

        if(isEmpty()){

            cout<<"Stack is Empty ";

        }

        else{

            cout<<"The peak element is: "<<arr[top]<<endl;

        }

    }

    void display(){

        if (isEmpty()) {

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

        } else {

            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 ---" << endl;

        cout << "1. Push" << endl;

        cout << "2. Pop" << endl;

        cout << "3. Peek" << endl;

        cout << "4. isEmpty" << endl;

        cout << "5. isFull" << endl;

        cout << "6. Display" << endl;

        cout << "7. Exit" << endl;

        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:

                if (s.isEmpty())

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

                else

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

                break;

            case 5:

                if (s.isFull())

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

                else

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

                break;

            case 6:

                s.display();

                break;

            case 7:

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

                break;

            default:

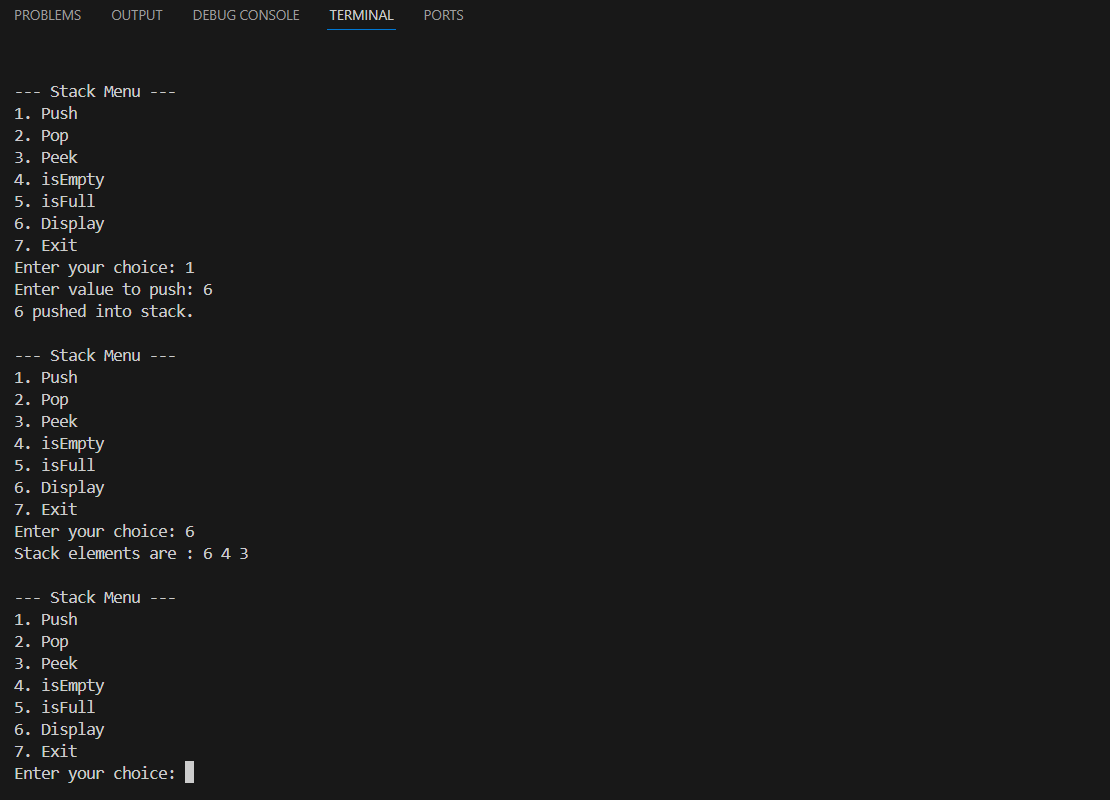
                cout << "Invalid choice! Try again." << endl;

        }

    } while (choice != 7);

    return 0;

}

Output: 

Ques2. Given a string, reverse it using STACK. For example “DataStructure” should be output as

“erutcurtSataD.”

Code: #include <iostream>

using namespace std;

class Stack {

private:

    char \*arr;

    int top;

    int capacity;

public:

    Stack(int size) {

        capacity = size;

        arr = new char[capacity];

        top = -1;

    }

    void push(char c) {

        if (top == capacity - 1) {

            cout << "Stack Overflow!" << endl;

            return;

        }

        arr[++top] = c;

    }

    char pop() {

        if (top == -1) {

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

            return '\0';

        }

        return arr[top--];

    }

    bool isEmpty() {

        return top == -1;

    }

};

int main() {

    string str;

    cout << "Enter a string: ";

    getline(cin, str);

    Stack s(str.length());

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

        s.push(str[i]);

    }

    string reversed = "";

    while (!s.isEmpty()) {

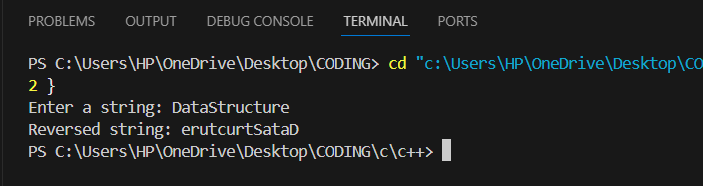
        reversed += s.pop();

    }

    cout << "Reversed string: " << reversed << endl;

    return 0;

}

Output: 

Ques3. Write a program that checks if an expression has balanced parentheses.

Code:

#include<iostream>

using namespace std;

const int MAX = 100;

class stack{

    private:

    int arr[MAX];

    int top;

    public:

    Stack() {

        top = -1;

     }

    bool isEmpty() {

         return top == -1;

        }

    bool isFull() {

        return top == MAX - 1;

    }

    void push(char ch) {

        if (isFull()) {

            cout << "Stack Overflow!" << endl;

            return;

        }

        else{

        arr[++top] = ch;

        }

    }

    char pop() {

        if (isEmpty()) {

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

            return '\0';

        }

        return arr[top--];

    }

    char peek() {

        if (!isEmpty())

            return arr[top];

        return '\0';

    }

};

bool isbalanced(string exp){

    stack s;

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

        char ch = exp[i];

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

            s.push(ch);

        }

         else if (ch == ')' || ch == '}' || ch == ']') {

            if (s.isEmpty())

                return false;

            char top = s.pop();

            if ((ch == ')' && top != '(') ||

                (ch == '}' && top != '{') ||

                (ch == ']' && top != '[')) {

                return false;

            }

        }

    }

    return s.isEmpty();

}

int main() {

    string exp;

    cout << "Enter an expression: ";

    cin >> exp;

    if (isbalanced(exp))

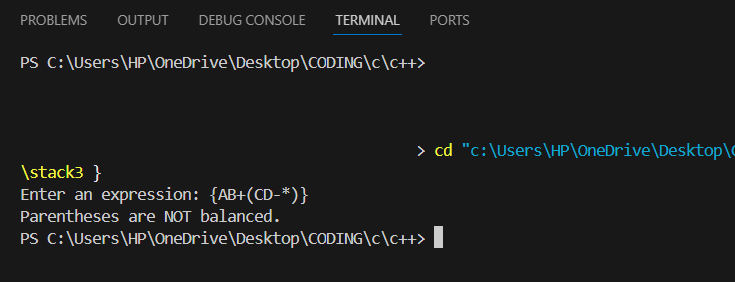
        cout << "Parentheses are balanced." << endl;

    else

        cout << "Parentheses are NOT balanced." << endl;

    return 0;

}

Output: 

Ques4. Write a program to convert an Infix expression into a Postfix expression.

Code:

#include <iostream>

#include <cstring>

using namespace std;

const int MAX = 100;

class Stack {

    private:

        char arr[MAX];

        int top;

    public:

        Stack() {

            top = -1;

        }

        bool isEmpty() {

            return top == -1;

        }

        bool isFull() {

            return top == MAX - 1;

        }

        void push(char c) {

            if (!isFull()) {

                arr[++top] = c;

            } else {

                cout << "Stack Overflow!" << endl;

            }

        }

        char pop() {

            if (!isEmpty()) {

                return arr[top--];

            } else {

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

                return '\0';

            }

        }

        char peek() {

            if (!isEmpty()) {

                return arr[top];

            } else {

                return '\0';

            }

        }

};

int precedence(char op) {

    if (op == '+' || op == '-') return 1;

    if (op == '\*' || op == '/') return 2;

    if (op == '^') return 3;

    return 0;

}

bool isOperator(char c) {

    return (c == '+' || c == '-' || c == '\*' || c == '/' || c == '^');

}

void infixToPostfix(char infix[]) {

    Stack s;

    char postfix[MAX];

    int k = 0;

    for (int i = 0; i < strlen(infix); i++) {

        char c = infix[i];

        // If operand, add to postfix

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

            postfix[k++] = c;

        }

        else if (c == '(') {

            s.push(c);

        }

        else if (c == ')') {

            while (!s.isEmpty() && s.peek() != '(') {

                postfix[k++] = s.pop();

            }

            s.pop();

        }

        else if (isOperator(c)) {

            while (!s.isEmpty() && precedence(s.peek()) >= precedence(c)) {

                postfix[k++] = s.pop();

            }

            s.push(c);

        }

    }

    while (!s.isEmpty()) {

        postfix[k++] = s.pop();

    }

    postfix[k] = '\0';

    cout << "Postfix Expression: " << postfix << endl;

}

int main() {

    char infix[MAX];

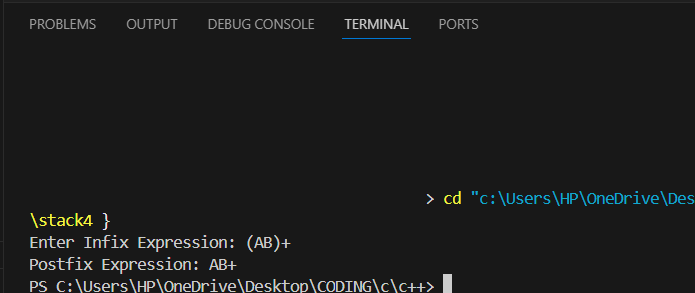
    cout << "Enter Infix Expression: ";

    cin >> infix;

    infixToPostfix(infix);

    return 0;

}

Output: 

Ques 5. Write a program for the evaluation of a Postfix expression.

Code:

#include <iostream>

#include <cstring>

#include <cctype>

using namespace std;

#define MAX 100

class Stack {

    int arr[MAX];

    int top;

public:

    Stack() { top = -1; }

    void push(int x) {

        if (top == MAX - 1) {

            cout << "Stack Overflow\n";

            return;

        }

        arr[++top] = x;

    }

    int pop() {

        if (top == -1) {

            cout << "Stack Underflow\n";

            return -1;

        }

        return arr[top--];

    }

    bool isEmpty() {

        return (top == -1);

    }

    int peek() {

        if (top == -1) {

            cout << "Stack Empty\n";

            return -1;

        }

        return arr[top];

    }

};

int evaluatePostfix(char exp[]) {

    Stack st;

    int len = strlen(exp);

    for (int i = 0; i < len; i++) {

        char ch = exp[i];

        // Skip spaces

        if (ch == ' ')

            continue;

        // If operand, push it

        if (isdigit(ch)) {

            st.push(ch - '0');  // convert char to int

        }

        else {

            // Operator case

            int val2 = st.pop();

            int val1 = st.pop();

            switch (ch) {

                case '+': st.push(val1 + val2); break;

                case '-': st.push(val1 - val2); break;

                case '\*': st.push(val1 \* val2); break;

                case '/': st.push(val1 / val2); break;

            }

        }

    }

    return st.pop();

}

int main() {

    char exp[MAX];

    cout << "Enter a postfix expression (single-digit operands): ";

    cin.getline(exp, MAX);

    cout << "Result = " << evaluatePostfix(exp) << endl;

    return 0;

}

Output:

