# Name memansi

# Roll no 1024030006

# Subgroup 2c35

# Assignment no 3

Develop a menu driven program demonstrating the following operations on a Stack using array: (i) push(), (ii) pop(), (iii) isEmpty(), (iv) isFull(), (v) display(), and (vi) peek().

#include <iostream>

using namespace std;

#define MAX 5

int stack[MAX], top = -1;

void push(int *x*) {

    if (top == MAX - 1)

        cout << "Stack Overflow!\n";

    else {

        stack[++top] = *x*;

        cout << *x* << " pushed.\n";

    }

}

void pop() {

    if (top == -1)

        cout<<"Stack Underflow\n";

    else

        cout<<stack[top--]<<"popped.\n";

}

void peek() {

    if (top == -1)

        cout<<"Stack is Empty\n";

    else

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

}

void display() {

    if (top == -1)

        cout <<"Stack is Empty\n";

    else {

        cout << "Stack: ";

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

            cout << stack[i] << " ";

        cout<<endl;

    }

}

int main() {

    int choice, value;

    do {

        cout<<"\n--- Stack Menu ---\n";

        cout<<"1. Push\n2. Pop\n3. isEmpty\n4. isFull\n5. Display\n6. Peek\n7. Exit\n";

        cout<<"Enter choice: ";

        cin>>choice;

        switch(choice) {

            case 1:

                cout<<"Enter value: ";

                cin>>value;

                push(value);

                break;

            case 2:pop();

            break;

            case 3:

                cout<<(top == -1 ? "Stack is Empty\n" : "Stack is NOT Empty\n");

                break;

            case 4:

                cout<<(top == MAX - 1 ? "Stack is Full\n" : "Stack is NOT Full\n");

                break;

            case 5:display();

            break;

            case 6:peek(); break;

            case 7:cout<<"Exiting\n";

             break;

            default:cout<<"Invalid choice!\n";

        }

    } while(choice != 7);

    return 0;

}



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

#include <iostream>

using namespace std;

int main() {

    char str[100], stack[100];

    int top = 1;

    cout<<"Enter a string: ";

    cin >> str;

*// push*

    for(int i=0;str[i]!='\0'; i++) {

        stack[++top] = str[i];

    }

    cout<<"Reversed string: ";

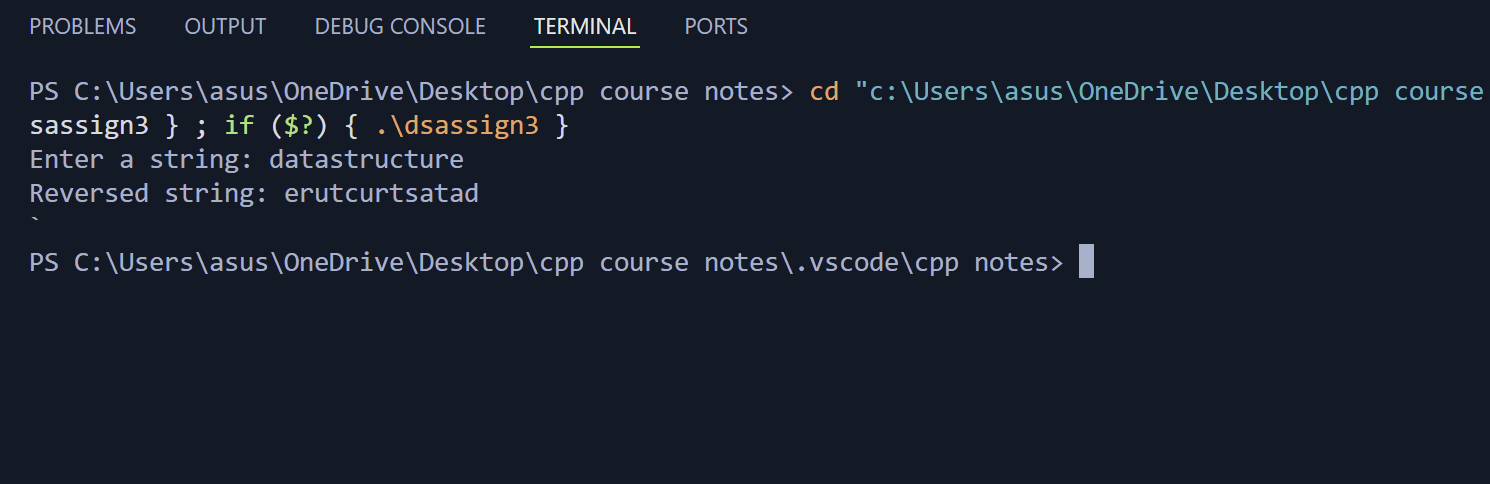
    while(top != -1) {

        cout<<stack[top--];

    }

    return 0;

}



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

#include <iostream>

using namespace std;

char stack[100];

int top = -1;

void push(char *c*) {

    stack[++top] = *c*;

}

char pop() {

    if (top == -1) return '\0';

    return stack[top--];

}

bool isBalanced(string *exp*) {

    for (int i = 0; *exp*[i] != '\0'; i++) {

        char ch = *exp*[i];

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

            push(ch);

        }

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

            if (top == -1) return false;

            char popped = pop();

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

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

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

                return false;

            }

        }

    }

    return (top == -1);

}

int main() {

    string expr;

    cout<<"Enter expression: ";

    cin>>expr;

    if(isBalanced(expr))

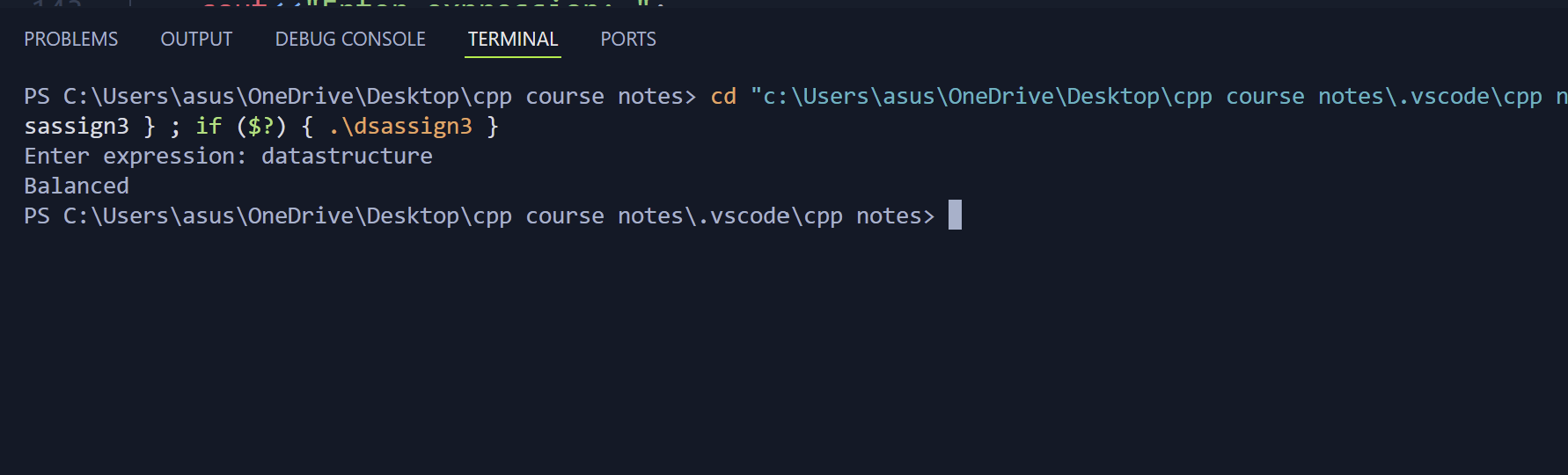
        cout<<"Balanced\n";

    else

        cout<<"Not Balanced\n";

    return 0;

}



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

#include <iostream>

#include <stack>

using namespace std;

int main() {

    string exp;

    cin>>exp;

    stack<int> s;

    for(char c:exp) {

        if(isdigit(c)) {

            s.push(c-'0');

        } else {

            int b=s.top();

             s.pop();

            int a=s.top();

            s.pop();

            if(c=='+') s.push(a+b);

            else if(c=='-') s.push(a-b);

            else if(c=='\*') s.push(a\*b);

            else if(c=='/') s.push(a/b);

        }

    }

    cout<<s.top();

    return 0;

}

