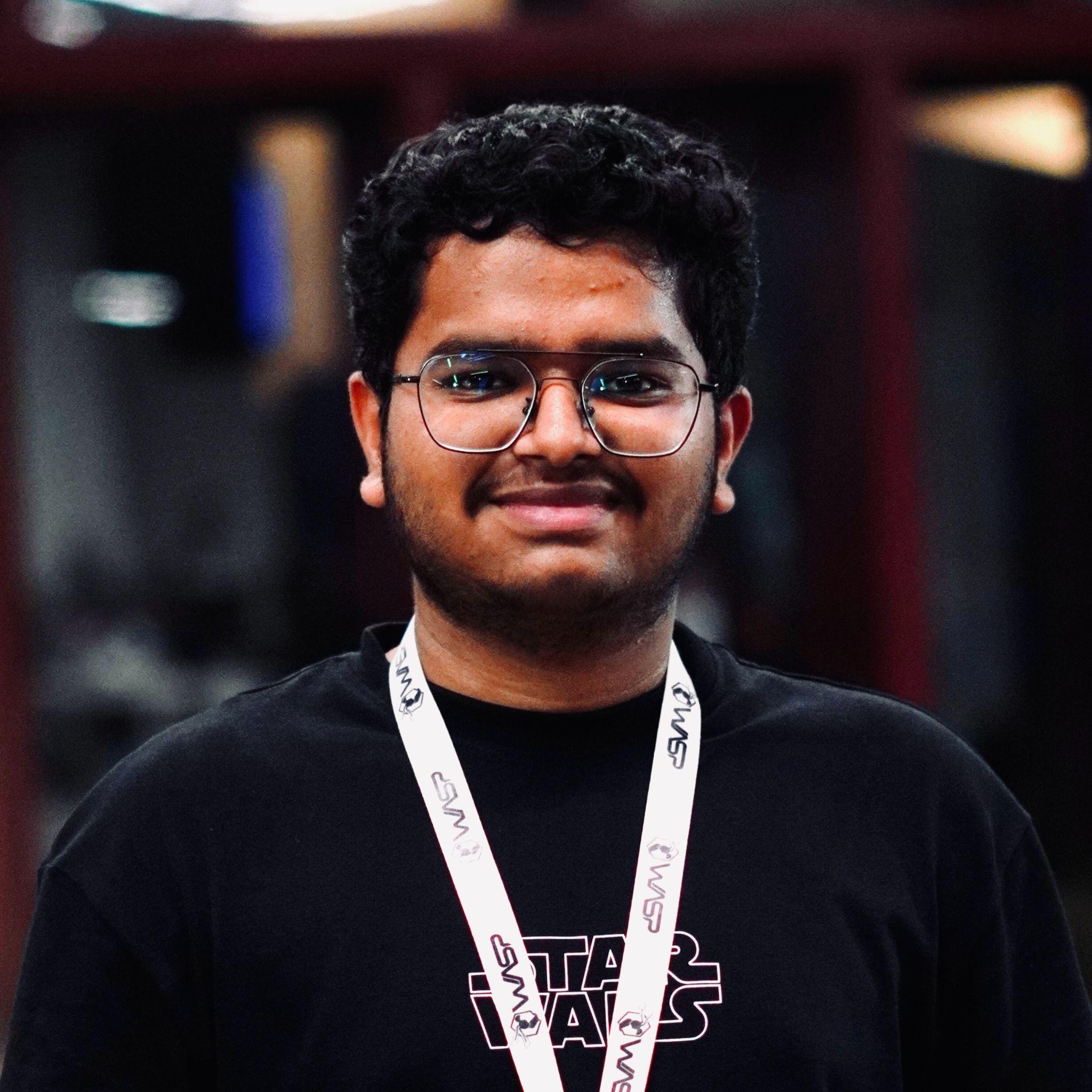
Name: Dhruv Mittal

Roll no: 1024030364

Subgroup: 2C24



**Q-1 :**

/\*

Menu driven program demonstrating Stack operations using array:

(i) push(), (ii) pop(), (iii) isEmpty(), (iv) isFull(), (v) display(), (vi) peek()

\*/

#include <iostream>

#include <climits>

using namespace std;

class Stack {

public:

int \*arr;

int top;

int size;

Stack(int size) {

this->size = size;

arr = new int[size];

top = -1;

}

~Stack() { // destructor to free memory

delete[] arr;

}

void push(int element) {

if (top < size - 1) {

arr[++top] = element;

} else {

cout << "Stack OverFlow" << endl;

}

}

void pop() {

if (top >= 0) {

cout << "Popped: " << arr[top--] << endl;

} else {

cout << "Stack UnderFlow" << endl;

}

}

int peek() {

if (isEmpty()) {

cout << "Stack is Empty" << endl;

return INT\_MIN; // return a sentinel value

}

return arr[top];

}

bool isEmpty() {

return top == -1;

}

bool isFull() {

return top == size - 1;

}

void display() {

if (isEmpty()) {

cout << "Stack is Empty" << endl;

} else {

cout << "Stack (top → bottom): ";

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

cout << arr[i] << " ";

}

cout << endl;

}

}

};

int main() {

int n;

cout << "Enter stack size: ";

cin >> n;

Stack s(n);

int choice, val;

do {

cout << "\n1. Push\n2. Pop\n3. isEmpty\n4. isFull\n5. Display\n6. Peek\n0. Exit\n";

cout << "Enter choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value: ";

cin >> val;

s.push(val);

break;

case 2:

s.pop();

break;

case 3:

cout << (s.isEmpty() ? "Yes, stack is empty\n" : "No, stack is not empty\n");

break;

case 4:

cout << (s.isFull() ? "Yes, stack is full\n" : "No, stack is not full\n");

break;

case 5:

s.display();

break;

case 6:

val = s.peek();

if (val != INT\_MIN)

cout << "Top element: " << val << endl;

break;

case 0:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice!\n";

}

} while (choice != 0);

return 0;

}

**Q-2 :**

/\*

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

“erutcurtSataD.”

\*/

#include <iostream>

#include <stack>

using namespace std;

int main() {

string str ="DataStructure";

stack<char> s;

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

char ch = str[i];

s.push(ch);

}

cout << "Reversed: ";

while (!s.empty()) {

cout << s.top();

s.pop();

}

cout << endl;

return 0;

}

**Q-3 :**

/\*

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

\*/

#include <iostream>

#include <stack>

using namespace std;

bool isBalanced(string str) {

stack<char> s;

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

char c = str[i];

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 str;

cout << "Enter expression: ";

cin >> str;

cout << (isBalanced(str) ? "Balanced" : "Not Balanced") << endl;

return 0;

}

**Q-4 :**

/\*

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

\*/

#include <iostream>

#include <stack>

using namespace std;

int priority(char c)

{

if (c == '^')

return 3;

if (c == '\*' || c == '/')

return 2;

if (c == '+' || c == '-')

return 1;

return -1;

}

string infixtopostfix(string s)

{

int n = s.length();

stack<char> st;

string ans = "";

for (int i = 0; i < n; i++)

{

if ((s[i] >= 'A' && s[i] <= 'Z') || (s[i] >= 'a' && s[i] <= 'z') || (s[i] >= '0' && s[i] <= '9'))

{

ans += s[i];

}

else if (s[i] == '(')

{

st.push(s[i]);

}

else if (s[i] == ')')

{

while (!st.empty() && st.top() != '(')

{

ans += st.top();

st.pop();

}

if (!st.empty())

st.pop();

}

else

{

while (!st.empty() && priority(s[i]) <= priority(st.top()))

{

ans += st.top();

st.pop();

}

st.push(s[i]);

}

}

while (!st.empty())

{

ans += st.top();

st.pop();

}

return ans;

}

int main()

{

string str;

cout << "Enter Infix: ";

cin >> str;

cout << "Postfix: " << infixtopostfix(str) << endl;

return 0;

}

**Q-5 :**

/\*

Write a program for the evaluation of a Postfix expression.

\*/

#include <iostream>

#include <stack>

#include <math.h>

using namespace std;

int evaluatePostfix(string str) {

stack<int> s;

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

char ch = str[i];

if (isdigit(ch)) s.push(ch - '0');

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;

}

}

}

return s.top();

}

int main() {

string str;

cout << "Enter Postfix: ";

cin >> str;

cout << "Evaluation: " << evaluatePostfix(str) << endl;

return 0;

}