**NAME:SAYALI JIVAN CHAUDHARI**

**ROLL NO.:14**

**PRN NO.2023015400005055**

**1)implementation of polish notation in cpp**

//infix to postfix

#include <bits/stdc++.h>

using namespace std;

// Function to return precedence of operators

int prec(char c) {

if (c == '^')

return 3;

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

return 2;

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

return 1;

else

return -1;

}

// Function to return associativity of operators

char associativity(char c) {

if (c == '^')

return 'R';

return 'L'; // Default to left-associative

}

// The main function to convert infix expression

// to postfix expression

void infixToPostfix(string s) {

stack<char> st;

string result;

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

char c = s[i];

// If the scanned character is

// an operand, add it to the output string.

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

result += c;

// If the scanned character is an

// ‘(‘, push it to the stack.

else if (c == '(')

st.push('(');

// If the scanned character is an ‘)’,

// pop and add to the output string from the stack

// until an ‘(‘ is encountered.

else if (c == ')') {

while (st.top() != '(') {

result += st.top();

st.pop();

}

st.pop(); // Pop '('

}

// If an operator is scanned

else {

while (!st.empty() && prec(s[i]) < prec(st.top()) ||

!st.empty() && prec(s[i]) == prec(st.top()) &&

associativity(s[i]) == 'L') {

result += st.top();

st.pop();

}

st.push(c);

}

}

// Pop all the remaining elements from the stack

while (!st.empty()) {

result += st.top();

st.pop();

}

cout << result << endl;

}

// Driver code

int main() {

string exp = "a+b\*(c^d-e)^(f+g\*h)-i";

// Function call

infixToPostfix(exp);

return 0;

}