Task-1

#include <iostream>

#include <string>

using namespace std;

struct Node

{

int data;

Node\* next;

};

class stack {

Node\* top;

public:

stack() {

top = NULL;

}

bool isempty() {

if (top == NULL) {

return true;

}

else

return false;

}

void push(char val) {

Node\* NewNode = new Node;

NewNode->data = val;

NewNode->next = NULL;

if (top != NULL) {

top->next = NewNode;

}

else

top = NewNode;

}

char pop() {

char ret = top->data;

Node\* temp = top;

top = top->next;

delete temp;

temp = NULL;

return ret;

}

char topp() {

if(top!=NULL)

return top->data;

return;

}

int precedence(char oper) {

if (oper == '^')

return 3;

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

return 2;

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

return 1;

else

return -1;

}

};

void infixToPostfix(string s)

{

stack st;

string result;

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

char c = s[i];

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

|| (c >= '0' && c <= '9'))

result += c;

else if (c == '(')

st.push('(');

else if (c == ')') {

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

result += st.topp();

st.pop();

}

st.pop();

}

else {

while (!st.isempty()

&& st.precedence(s[i]) <= st.precedence(st.topp())) {

result += st.topp();

st.pop();

}

st.push(c);

}

}

while (!st.isempty()) {

result += st.topp();

st.pop();

}

cout << result << endl;

}

int main() {

string str;

cout << "Enter a string: ";

str = "(6+2)\*5-8/4";

cout << str << endl;

infixToPostfix(str);

system("pause");

return 0;

}

Task-2