Name – Yashwant Patidar

Scholar Number – 191112243

Section – CSE\_2

Subject – Data Structures lab

**Ques1.** WAP to convert inflix expression to postfix expression.

:-

#include <iostream>

#include <string>

using namespace std;

struct node

{

char data;

node \*next;

};

class stack

{

node \*root;

public:

void push(char data);

char pop();

char top();

};

void stack::push(char data)

{

node \*stackNode = new node();

stackNode->data = data;

stackNode->next = root;

root = stackNode;

}

char stack::pop()

{

if (!root)

return '\0';

node \*temp = root;

root = root->next;

char popped = temp->data;

free(temp);

return popped;

}

char stack::top()

{

if (!root)

return '\0';

return root->data;

}

int prec(char c)

{

if (c == '^')

return 3;

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

return 2;

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

return 1;

else

return -1;

}

int main()

{

stack st;

string s, ns;

int i;

cout << "Enter the Expression: ";

getline(cin, s);

s += ')';

st.push('(');

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

{

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

ns += s[i];

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

st.push('(');

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

{

while (st.top() != '\0' && st.top() != '(')

{

char c = st.top();

st.pop();

ns += c;

}

if (st.top() == '(')

{

char c = st.top();

st.pop();

}

}

else

{

while (st.top() != '\0' && prec(s[i]) <= prec(st.top()))

{

char c = st.top();

st.pop();

ns += c;

}

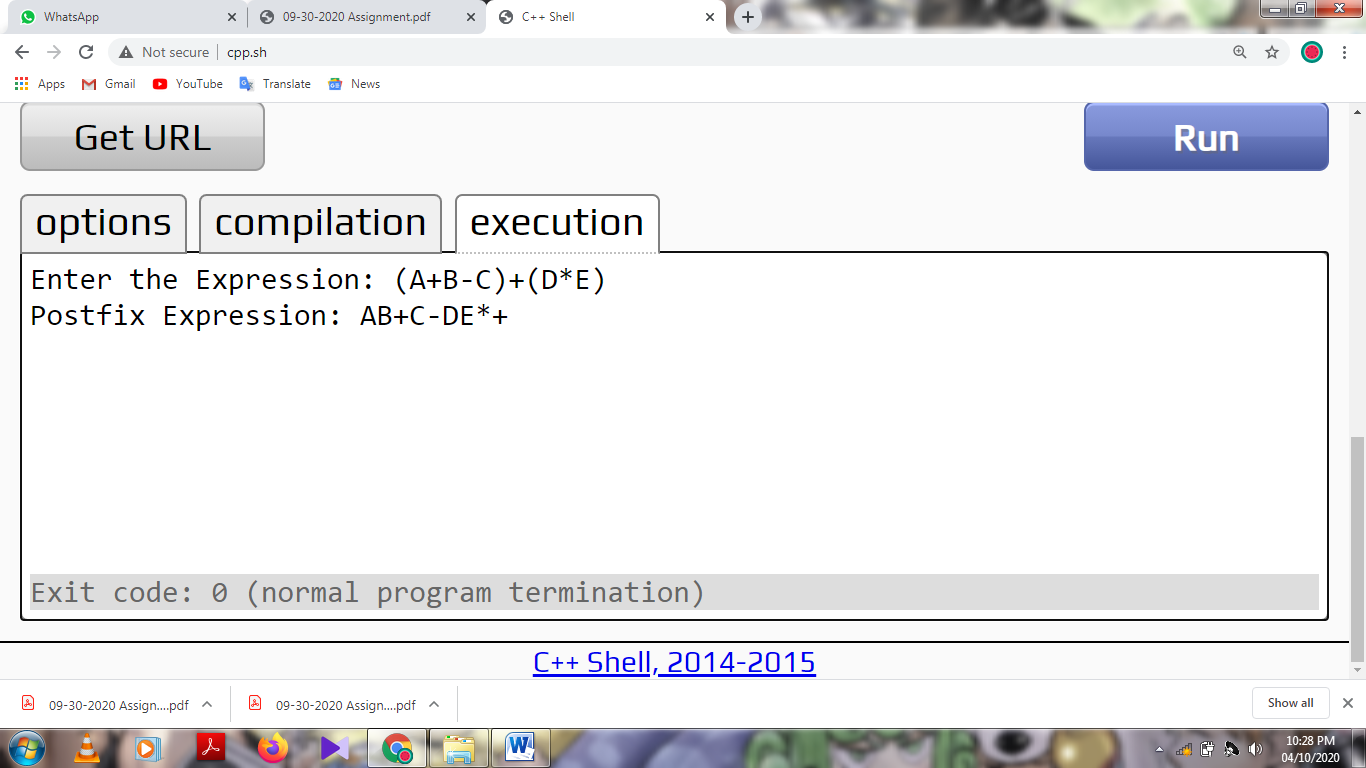
st.push(s[i]);

}

}

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

}



**Ques2.** WAP to convert infix expression to prefix expression.

:-

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

struct node

{

char data;

node \*next;

};

class stack

{

node \*root;

public:

void push(char data);

char pop();

char top();

};

void stack::push(char data)

{

node \*stackNode = new node();

stackNode->data = data;

stackNode->next = root;

root = stackNode;

}

char stack::pop()

{

if (!root)

return '\0';

node \*temp = root;

root = root->next;

char popped = temp->data;

free(temp);

return popped;

}

char stack::top()

{

if (!root)

return '\0';

return root->data;

}

int prec(char c)

{

if (c == '^')

return 3;

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

return 2;

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

return 1;

else

return -1;

}

int main()

{

stack st;

string s, ns;

int i;

cout << "Enter the Expression: ";

getline(cin, s);

s += ')';

reverse(s.begin(), s.end());

s += '(';

st.push('(');

for (auto &ch : s)

{

if (ch == '(')

ch = ')';

else if (ch == ')')

ch = '(';

}

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

{

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

ns += s[i];

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

st.push('(');

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

{

while (st.top() != '\0' && st.top() != '(')

{

char c = st.top();

st.pop();

ns += c;

}

if (st.top() == '(')

{

char c = st.top();

st.pop();

}

}

else

{

while (st.top() != '\0' && prec(s[i]) <= prec(st.top()))

{

char c = st.top();

st.pop();

ns += c;

}

st.push(s[i]);

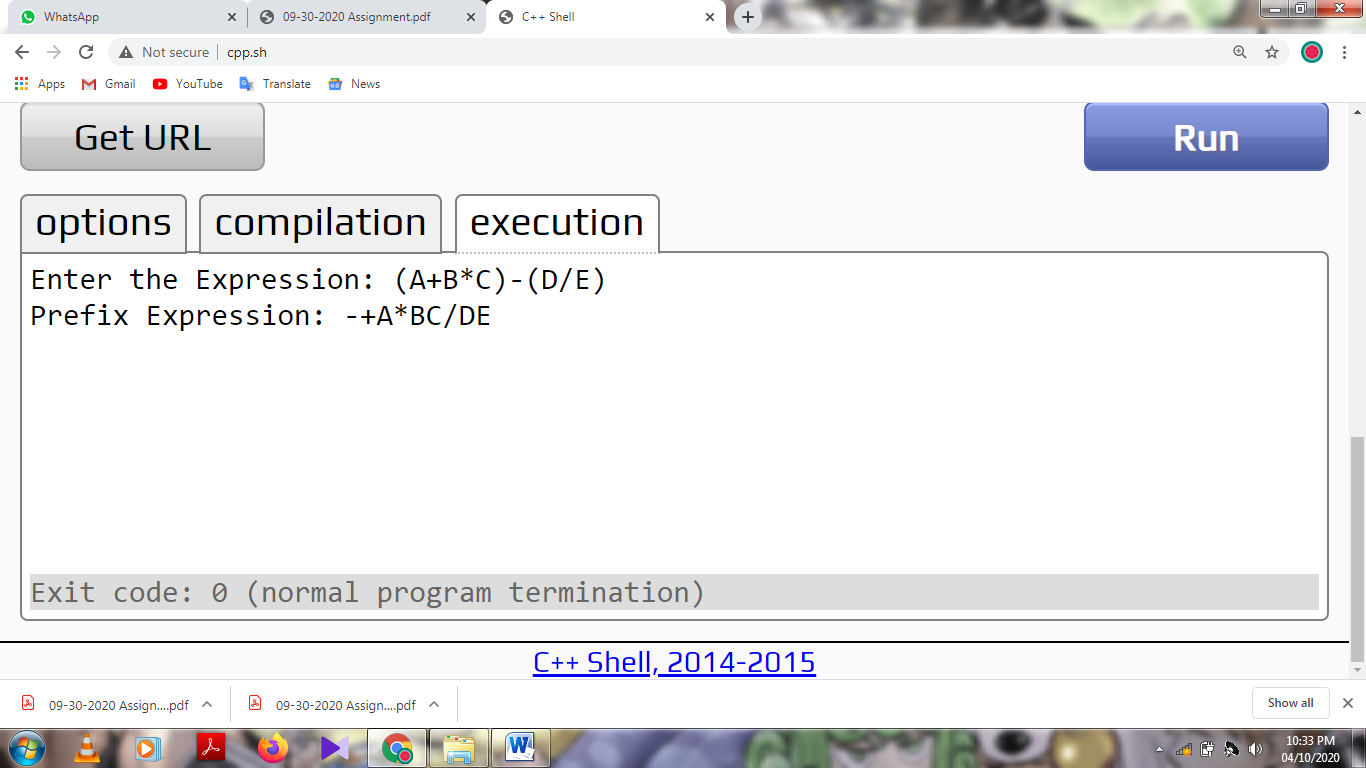
}

}

reverse(ns.begin(), ns.end());

cout << "Prefix Expression: " << ns << endl;

}



**Ques3.** WAP to evaluate postfix expression.

:-

#include <iostream>

#include <cmath>

#include <climits>

#include <string>

using namespace std;

struct node

{

float data;

node \*next;

};

class stack

{

node \*root;

public:

void push(float data);

float pop();

float top();

};

void stack::push(float data)

{

node \*stackNode = new node();

stackNode->data = data;

stackNode->next = root;

root = stackNode;

}

float stack::pop()

{

if (!root)

return '\0';

node \*temp = root;

root = root->next;

float popped = temp->data;

free(temp);

return popped;

}

float stack::top()

{

if (!root)

return '\0';

return root->data;

}

float scanNum(char ch)

{

int value;

value = ch;

return float(value - '0');

}

int isOperator(char ch)

{

switch (ch)

{

case '+':

case '-':

case '\*':

case '/':

case '^':

return 1;

}

return -1;

}

int isOperand(char ch)

{

if (ch >= '0' && ch <= '9')

return 1;

return -1;

}

float operation(int a, int b, char op)

{

switch (op)

{

case '+':

return b + a;

case '-':

return b - a;

case '\*':

return b \* a;

case '/':

return b / a;

case '^':

return pow(b, a);

}

return INT\_MIN;

}

int main()

{

int a, b, i;

stack stk;

string str;

cout << "Enter the Expression: ";

getline(cin, str);

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

{

if (isOperator(str[i]) != -1)

{

a = stk.pop();

b = stk.pop();

stk.push(operation(a, b, str[i]));

}

else if (isOperand(str[i]))

{

int j = 0;

float num = 0;

while (str[i + j] != '!')

{

int value = str[i + j];

num = float(value - '0') + (num \* pow(10, j++));

}

i += j;

stk.push(num);

}

}

cout << "Answer: " << stk.top() << endl;

}

