Practical No:- 3.(b)

1. (b) write a program to convert an infix expression to post fix and prefix conversion . convert infix expression to the post fix notation.

Practical implementation:-

Code:-

// CPP program to convert infix to prefix

#include <bits/stdc++.h>

using namespace std;

bool isOperator(char c)

{

return (!isalpha(c) && !isdigit(c));

}

int getPriority(char C)

{

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

return 1;

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

return 2;

else if (C == '^')

return 3;

return 0;

}

string infixToPostfix(string infix)

{

infix = '(' + infix + ')';

int l = infix.size();

stack<char> char\_stack;

string output;

for (int i = 0; i < l; i++) {

// If the scanned character is an

// operand, add it to output.

if (isalpha(infix[i]) || isdigit(infix[i]))

output += infix[i];

// If the scanned character is an

// ‘(‘, push it to the stack.

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

char\_stack.push('(');

// If the scanned character is an

// ‘)’, pop and output from the stack

// until an ‘(‘ is encountered.

else if (infix[i] == ')') {

while (char\_stack.top() != '(') {

output += char\_stack.top();

char\_stack.pop();

}

// Remove '(' from the stack

char\_stack.pop();

}

// Operator found

else

{

if (isOperator(char\_stack.top()))

{

if(infix[i] == '^')

{

while (getPriority(infix[i]) <= getPriority(char\_stack.top()))

{

output += char\_stack.top();

char\_stack.pop();

}

}

else

{

while (getPriority(infix[i]) < getPriority(char\_stack.top()))

{

output += char\_stack.top();

char\_stack.pop();

}

}

// Push current Operator on stack

char\_stack.push(infix[i]);

}

}

}

while(!char\_stack.empty()){

output += char\_stack.top();

char\_stack.pop();

}

return output;

}

string infixToPrefix(string infix)

{

/\* Reverse String

\* Replace ( with ) and vice versa

\* Get Postfix

\* Reverse Postfix \* \*/

int l = infix.size();

// Reverse infix

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

// Replace ( with ) and vice versa

for (int i = 0; i < l; i++) {

if (infix[i] == '(') {

infix[i] = ')';

}

else if (infix[i] == ')') {

infix[i] = '(';

}

}

string prefix = infixToPostfix(infix);

// Reverse postfix

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

return prefix;

}

// Driver code

int main()

{

string s = ("x+y\*z/w+u");

cout << infixToPrefix(s) << std::endl;

return 0;

}

Output:-

