**Compiler Design**

**Experiment - 10**

**Sai Ram Pendyala**

**RA1911003010696**

**Postfix Evaluation**

**Aim:**

To evaluate a postfix expression.

**Algorithm:**

1) Create a stack to store operands (or values).

2) Scan the given expression and do the following for every scanned element.

…..a) If the element is a number, push it into the stack

…..b) If the element is an operator, pop operands for the operator from the stack. Evaluate the operator and push the result back to the stack

3) When the expression is ended, the number in the stack is the final answer

**Program:**

#include <bits/stdc++.h>

using namespace std;

int evaluatePostfix(string exp) {

stack<int> st;

for(int i = 0; exp[i] != '\0'; i++) {

if(isdigit(exp[i]))

st.push(exp[i]-'0');

else{

int op2 = st.top();

st.pop();

int op1 = st.top();

st.pop();

switch (exp[i]) {

case '+': st.push(op1+op2); break;

case '-': st.push(op1-op2); break;

case '\*': st.push(op1\*op2); break;

case '/': st.push(op1/op2); break;

case '^': st.push(op1^op2); break;

}

}

}

return st.top();

}

int main()

{

string exp;

cout << "Postfix expression: ";

cin >> exp;

cout << "Result: ";

cout << evaluatePostfix(exp);

return 0;

}

**Output:**



**Result:**

Hence, the given postfix expression is evaluated.

**Prefix Evaluation**

**Aim:**

To evaluate a postfix expression.

**Algorithm:**

Step 1: Put a pointer P at the end of the end

Step 2: If character at P is an operand push it to Stack

Step 3: If the character at P is an operator pop two

elements from the Stack. Operate on these elements

according to the operator, and push the result

back to the Stack

Step 4: Decrement P by 1 and go to Step 2 as long as there

are characters left to be scanned in the expression.

Step 5: The Result is stored at the top of the Stack,

return it

Step 6: End

**Program:**

#include <bits/stdc++.h>

using namespace std;

int evaluatePrefix(string exp) {

stack<int> st;

for(int i = exp.size()-1; i >= 0; i--) {

if(isdigit(exp[i]))

st.push(exp[i]-'0');

else {

int op1 = st.top();

st.pop();

int op2 = st.top();

st.pop();

switch (exp[i]) {

case '+': st.push(op1+op2); break;

case '-': st.push(op1-op2); break;

case '\*': st.push(op1\*op2); break;

case '/': st.push(op1/op2); break;

case '^': st.push(op1^op2); break;

}

}

}

return st.top();

}

int main()

{

string exp;

cout << "Prefix expression: ";

cin >> exp;

cout << "Result: ";

cout << evaluatePrefix(exp);

return 0;

}

**Output:**



**Result:**

Hence, the given prefix expression is evaluated.