Ex. No. 4b **Infix To Postfix Conversion**

Date:

**Aim**

To convert infix expression to its postfix form using stack operations.

**Algorithm**

1. Start

2. Define a array stack of size max = 20

3. Initialize top = -1

4. Read the infix expression character-by-character

If character is an operand print it

If character is an operator

Compare the operator’s priority with the stack[top] operator.

If the stack [top] has higher/equal priority than the input operator,

Pop it from the stack and print it.

Else

Push the input operator onto the stack

If character is a left parenthesis, then push it onto the stack.

If character is a right parenthesis, pop all operators from stack and print

it until a left parenthesis is encountered. Do not print the parenthesis.

If character = $ then Pop out all operators, Print them and Stop

**Program**

/\* Conversion of infix to postfix expression \*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 20

int top = -1;

char stack[MAX];

char pop();

void push(char item);

int prcd(char symbol)

{

switch(symbol)

{

case '+':

case '-':

return 2;

break;

case '\*':

case '/':

return 4;

break;

case '^':

case '$':

return 6;

break;

case '(':

case ')':

case '#':

return 1;

break;

}

}

int isoperator(char symbol)

{

switch(symbol)

{

case '+':

case '-':

case '\*':

case '/':

case '^':

case '$':

case '(':

case ')':

return 1;

break;

default:

return 0;

}

}

void convertip(char infix[],char postfix[])

{

int i,symbol,j = 0;

stack[++top] = '#';

for(i=0;i<strlen(infix);i++)

{

symbol = infix[i];

if(isoperator(symbol) == 0)

{

postfix[j] = symbol;

j++;

}

else

{

if(symbol == '(')

push(symbol);

else if(symbol == ')')

{

while(stack[top] != '(')

{

postfix[j] = pop();

j++;

}

pop(); //pop out (.

}

else

{

if(prcd(symbol) > prcd(stack[top]))

push(symbol);

else

{

while(prcd(symbol) <= prcd(stack[top]))

{

postfix[j] = pop();

j++;

}

push(symbol);

}

}

}

}

while(stack[top] != '#')

{

postfix[j] = pop();

j++;

}

postfix[j] = '\0';

}

main()

{

char infix[20],postfix[20];

system("clear");

printf("Enter the valid infix string: ");

fgets(infix,20,stdin);

convertip(infix, postfix);

printf("The corresponding postfix string is: ");

puts(postfix);

}

void push(char item)

{

top++;

stack[top] = item;

}

char pop()

{

char a;

a = stack[top];

top--;

return a;

}

**Output**

**Result**

Thus the given infix expression was converted into postfix form using stack.