## Procedure to convert from infix expression to postfix expression is as follows:

## 1. Scan the infix expression from left to right.

## 2. a) If the scanned symbol is left parenthesis, push it onto the stack.

## b) If the scanned symbol is an operand, then place directly in the postfix expression

## (output).

## c) If the symbol scanned is a right parenthesis, then go on popping all the items from

## the stack and place them in the postfix expression till we get the matching left

## parenthesis.

## d) If the scanned symbol is an operator, then go on removing all the operators from

## the stack and place them in the postfix expression, if and only if the precedence of

## the operator which is on the top of the stack is greater than (or greater than or

## equal) to the precedence of the scanned operator and push the scanned operator

## onto the stack otherwise, push the scanned operator onto the stack.

## /\* C Program to Convert Infix to Postfix using Stack\*/

#include<stdio.h>

#include<ctype.h>

char stack[100];

int top = -1;

void push(char x)

{

stack[++top] = x;

}

char pop()

{

if(top == -1)

return -1;

else

return stack[top--];

}

int priority(char x)

{

if(x == '(')

return 0;

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

return 1;

if(x == '\*' || x == '/')

return 2;

return 0;

}

int main()

{

char exp[100];

char \*e, x;

printf("Enter the expression : ");

scanf("%s",exp);

printf("\n");

e = exp;

while(\*e != '\0')

{

if(isalnum(\*e))

printf("%c ",\*e);

else if(\*e == '(')

push(\*e);

else if(\*e == ')')

{

while((x = pop()) != '(')

printf("%c ", x);

}

else

{

while(priority(stack[top]) >= priority(\*e))

printf("%c ",pop());

push(\*e);

}

e++;

}

while(top != -1)

{

printf("%c ",pop());

}

return 0;

}

Output

**Case 1:**

Enter the expression : a+b\*c

a b c \*+

**Case 2**

Enter the expression : (a+b)\*c+(d-a)

a b + c \* d a -+

**Case 3**

Enter the expression : ((4+8)(6-5))/((3-2)(2+2))

4 8 + 6 5 - 3 2 - 2 2 + /