Infix to Postfix conversion

**ALGORITHM**

**Input**: Read an Infix expression

**Output**: Postfix expression

**Steps**:

Declare char array stack[20] and int variable top=-1

**push(char x)**

1. Start
2. Stack[++top]=x
3. Stop

**pop( )**

1. Start
2. if(top == -1)
3. return -1
4. else

return stack[top--]

1. Stop

**priority(char x)**

1. Start
2. if(x == '+' || x == '-')
3. return 1
4. Else if(x == '\*' || x == '/')
5. return 2
6. Else

return 0

1. Stop

**main( )**

1. Start
2. Declare char array exp, char pointer e and char variable x
3. Print "Enter the expression : "
4. Read exp
5. e = exp
6. while(\*e != '\0')
7. if(isalnum(\*e))
8. Print \*e
9. else if(\*e == '(')
10. push(\*e)
11. else if(\*e == ')')
12. while((x = pop( )) != '(')
13. Print x
14. else
15. while(priority(stack[top]) >= priority(\*e))
16. Print pop()
17. push(\*e)
18. e++
19. Stop

**PROGRAM**

#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 == '+' || x == '-')

return 1;

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

return 2;

else if(x==’^’)

return 3

else

return 0;}

void 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());

}

}