LAB 3 - Infix to Postfix.

WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators

+, -, *, / .

```
#include <stdio.h>
#define MAX 100
char stack[MAX];
int top = -1;

void push(char ch);
char pop();
int stackempty();
char stacktop();
int priority(char ch);

int main()
{
    char infix[100];
    int i, item;
    printf("Enter the infix expression : ");
    scanf("%s", infix);
    printf("\n postfix : ");
```

```
i = 0;
while (infix[i] != '\0')
{
    switch (infix[i])
    {
        case '(' : push (infix[i]);
                   break;
        case ')' : while ((item = pop()) != '(')
                       printf ("%c", item);
                   break;
        case '+' :
        case '-' :
        case '*' :
        case '/' :
        case '^' :
                   while (! stackempty() &&
                       priority (infix[i]) <=
                                priority (stacktop())
                   {
                       item = pop();
                       printf ("%c", item);
                   }
                   push (infix[i]);
                   break;
```

```
        default : printf("%c", infix[i]);
                  break;
    }
    i++;
}

while (!stackempty())
{
    char item;
    item = pop();
    printf("%c", item);
}
printf("\n");
return 0;
}

void push(char ch)
{
    if (top = MAX-1)
        printf("Stack is full \n");
    else
    {
        top++;
        stack[top] = ch;
    }
}
```

```
char pop()
{
    char item;
    if (top == -1)
        printf("\n stack is empty !");
    else
    {
        item = stack[top];
        top--;
        return item;
    }
}

int stackempty()
{
    if (top == -1)
    return 1;
    else
    return 0;
}

char stacktop()
{
    if (top ==-1)
    printf("\n stack is empty !");
    return '\0';
    else
    return stack[top];
}
```

```
int priority (char ch)
{
    switch (ch)
    {
        case '+':
        case '-': return (1);
        case '*':
        case '/': return (2);
        case '^': return (3);
        default : return (0);
    }
}
```

expected output.

Enter the infix expression : 6*(5+(2+3)*8+3)

Postfix : 65234+8*+3+*