Lab Program - 2

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define size 100

char check(size);
int top = -1;

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

char pop ()
{
   if (top == -1)
     printf (" Stack UNDERFLOW! Stack is empty\n");
```

```
    else {
        char item = stack[top];
        top--;
        return item;
    }
}

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

char stacktop()
{
    if(top==-1)
        printf("\n Stack Underflow! Stack is Empty!");
    else
        return stack[top];
}

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

```
int main()
{
    char infix[size];
    int i, itemn;
    printf("Enter a VALID infix expression :\n");
    scanf("%s", infix);
    for(int i=0; i<strlen(infix); i++)
    {
        if((infix[i] == '*' || infix[i] == '+' ||
            infix[i] == '/' || infix[i] == '-' ||
            infix[i] == '^' || infix[i] == '(')
            && (infix[i+1] == '*' || infix[i+1] == '-'
            infix[i+1] == '+' || infix[i+1] == '/' ||
            infix[i+1] == '^' || infix[i+1] == ')'))
        {
            printf("INVALID EXPRESSION\n");
            exit(0);
        }
    }
    printf("The entered Infix Expression is : %s",
                                        infix);
    printf("\nThe generated Postfix Expression is :");
    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;
}
```