Exp 03

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>

char STACK[100];
int top=-1;
void push(char);
char pop();
int is_op(char);
int precedence(char);
void infix_to_postfix(char infix[],char postfix[]);

int main()
{
	char infix[100], postfix[100];
	printf("Enter infix expression:\n");
	gets(infix);

	infix_to_postfix(infix, postfix);
	printf("\nPostfix expression is:\n");
	puts(postfix);
	return 0;
}

void push(char ITEM)
{
```

```
	if(top>=99)
		printf("Stack Overflow");
	else
	{
		top++;
		ITEM=STACK[top];
	}
}

char pop()
{
	char item;
	if (top<0)
	{
		printf("Stack Underflow");
		getchar();

		exit(1);
	}
	else
	{
		item=STACK[top];
		top--;
		return  (item);
	}
}

int is_op(char ITEM)	//return true if ITEM is an operator, else return 0
{
	if (ITEM=='+'||ITEM=='-'||ITEM=='/'||ITEM=='*'||ITEM=='^')
		return 1;
```

```
	else
		return 0;
}

int precedence( char ITEM)  //gives precedence of ITEM. ^=3, '/' & '*'=2, and '+' & '-' =1, rest
0
{
	if(ITEM=='^')
		return 3;
	else if(ITEM=='*'||ITEM=='/')
		return 2;
	else if(ITEM=='+'||ITEM=='-')
		return 1;
	else
		return 0;
}

void infix_to_postfix(char infix[],char postfix[])
{
	char x, ITEM;
	int i=0,j=0;

	push('(');		//Push '(' in STACK
	strcat(infix,")");	// add ')' to infix

	ITEM=infix[0];

	while (ITEM != '\0')
	{
		if(ITEM == '(')
		{	push(ITEM);
```

```
        }
else if(is_op(ITEM)==1)
{
        x=pop();
        while(is_op(x)==1 && precedence(x)>= precedence(ITEM))
        {
                postfix[i]=x;
                i++;
                x=pop();
        }
        push(x);
        push(ITEM);             //push current op symbol into STACK
}
else if( !isdigit(infix[i])  || !isalpha(infix[i]) )
{
        postfix[i]=ITEM;
        i++;
}

else if(ITEM==')')
{
        x=pop();
        while(x !='(')
        {
                postfix[i]=x;
                i++;
                x=pop();
        }
}
else
{
```

```
                        printf("\nInvalid Infix Expression___1");
                        getchar();
                        exit(1);
                }
                j++;
                ITEM=infix[j];
        }



        if (top<0)
        {
                printf("\nInvalid Infix Expression");
                getchar();
                exit(1);
        }
        postfix[i]='\0';
}
```

Output: