PROGRAM - 04

Design, develop and implement a program in C for converting an infix expression to postfix expression. Program should support for both parenthesized and free parenthesized expression with the operators: "+", "-", "*", "/", "%", "^" and alphanumeric operands.

```
#include<stdio.h>
#include<conio.h>
 int sp(symbol)
  {
    switch(symbol)
     {
      case '+' :
      case '-' : return 2;
      case '*' :
      case '/' : return 4;
      case '^' :
      case '$' : return 5;
      case '(' : return -1;
      default :  return 8;
     }
  }
  Int IP(symbol)
    {
     switch(symbol)
      {
```

command line

- To open / create a file -
  gedit <filename.c>
- To compile a program -
  gcc filename.c -lm.
- To see the output.
  ./a.out

Output

1) Enter a valid infix expression.
   a+b*c
   the infix exp is a+b*c
   the postfix exp is abc*+

D D M M Y Y Y Y
1 3 0 9 2 0 1 7

```
{
 case '+':
 case '-': return 1;
 case '*':
 case '/': return 3;
 case '^':
 case '$': return 5;
 case '(':
 case ')': return 9;
 default : return 7;
 }
}
void infix-postfix(char infix[], char postfix[])
{
 int top, i, j;
 char S[30], symbol;
 top = -1;
 S[++top] = '#'; j=0;
 for(i=0; i<strlen(infix); i++)
  {
   while(sp(S[top]) > IP(symbol))
    {
     postfix[j] = S[top--];
     j++;
    }
  }
 if(sp[sp[top] != IP(symbol))
     S[++top] = symbol;
```

D D M M Y Y Y Y
1 3 0 9 2 0 1 7

```
else
    top--;
}
while(s[top]!='#')
 {
  postfix[j++]=s[top--];
 }
postfix[j]='\0';
}
void main()
{
 char infix[20];
 postfix[20]; clrscr();
 printf("Enter a valid infix expression\n");
 gets(infix);
 infix_postfix(infix, postfix);
 printf("The infix expression is :\n");
 printf("%s", infix);
 printf("The postfix expression is \n");
 printf("%s", postfix);
 getch();
}
```

output

```
gedit sa.c.
gcc sa.c
./a.out.

Enter the postfix expression.
11+.
The result is : 2.000000
```

D D M M Y Y Y Y
1 3 0 9 2 0 1 7

PROGRAM-05

a) Design, Develop and Implement a program in C for the following stack applications.
Evaluation of suffix expression with single digit operands and operator: +, -, *, /, %, ^.

```
#include <stdio.h>
#include <math.h>
#include<string.h>
double compute( char symbol, double op1, double op2)
{
  switch(symbol)
   {
    case '+': return op1+op2;
    case '-': return op1-op2;
    case '*': return op1*op2;
    case '/': return op1/op2;
    case '%':
    case '^': return pow(op1, op2);
    default : return 0;
    }
  }
void main()
 {
   double S[20],res, op1, op2;
   int top, i;
   char postfix[20], symbol;
```

D D M M Y Y Y Y

```
printf("\n Enter the postfix expression\n");
gets(postfix);
top = -1;
for(i=0; i<strlen(postfix); i++)
{
  symbol = postfix[i];
  if(isdigit(symbol))
  s[++top] = symbol - '0';
 else
  {
    op2 = s[top--];
    op1 = s[top--];
   res = compute(symbol, op1, op2);
    s[++top] = res;
    }
  }
res = s[top--];
 printf("\n The result is %f \n", res);
 }
```