HEMANG SINGH
1BM19CS061

Practice - 2

1) Infix to Prefix

```
#include <stdio.h>
#include <string.h>
int F(char Symbol)
 {
  switch(symbol)
  {
   ~~s~~ case '+' :
     case '-' : return 1;
     case '*' :
     case '/' : return 3;
     case '^' :
     case '$' : return 6;
     case ~~'('~~ ')' : return 9;
     case '#' : return 0;
     default : return 8;

  }
 }
 int G(char symbol)
   {
    switch(symbol)
     {
      ~~case~~ '+' :
      case ~~$~~ '-' :   return 2;
      case '*' :
      case '/' :   return 4;
      case '^' :
      case '$' :   return 5;
      case '(' :   return 0;
      case ')' :   return 9;
```

```
        default : return 7;
}
}
void infix - prefix (char infix[], char prefix[])
 {
    int  top, i, j;
    char  s[30], symbol;
    top = -1;
    s[++top] = '#';
    j = 0;
    Strrev(infix);
    for(i=0 ; i<strlen(infix) ; j++)
    {
       symbol = infix [i];
       while (F(s[top]) != G(symbol))
        {
          prefix [j] = s[top--];
           j++;
        }
       if (F(s[top]) != G(symbol))
          s[++top] = symbol;
          else
          top--;
  }
  while ( s [top] != '#')
  {
     prefix [j++] = s [top --];
  }
  prefix [j] = '\0' $;
  Strrev (prefix);
 }
    int main()
```

```
{
  char infix [20] ;
  char infix [20];
   printf ("Enter the valid infix expression \n");
   scanf ("%s", infix);
   infix_prefix (infix, prefix);
   printf ("The prefix expression is : \n");
   printf (" %s \n", prefix);

}
```

HEMANG SINGH
1BM19CS061

Practice - 2

2) Demonstrate evaluation of postfix expression

```
# include < Stdio.h>
# include < math.h>
# include < String.h>
# include < ctype.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);
    }
}
void main()
{
    double s[20];
    double op1, op2;
    float res;
    int top, i;
    char postfix [20], symbol;
    printf ("enter the postfix expression \n");
    scanf ("%s", 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 (" result = %f \n ", res);
}
```

Practice - ~~1~~2

1BM19CS061

3) factorial of a number using Recursion.

```
# include <stdio.h>
int fact( int n)
{
    if (n = = 0)
       return 1;
       else
       return    n * fact (n-1);

 }
 void main()
  {
     int n;
     printf( " enter the value of n \n");
     scanf(" %d", &n);
     printf(" the factorial of %d = %d \n", n,
                                        fact(n));
}
```

HEMANG SINGH
IBM19CS061

Practice - 2

4) GCD of numbers using Recursion.

```
# include <stdio.h>
int gcd( int n, int m);
int main() {
        int n, m;
        printf(" Enter two positive integers : ");
        scanf(" %d %d", &n, &m);
        printf(" G. C.D of %d and %d is %d.",
                                  n, m, gcd (n, m));

        return 0;

    }
    int   gcd (int n , int m) {
          if (m != 0)
               return gcd (m, n % m);

          else
               return n ;

    }
```