Prg: C program to find factorial of n using Recursion.

1BM19CS090
RAHIL

```
#include < stdio.h >
~~#include < conio.h >~~
int fact (int n)
{
    if (n==0) return 1;
    return n * fact (n-1);
}
void main ()
{
    int n;
    ~~clrscr~~
    Printf (" Enter the value of n \n");
    scanf (" %d", &n);
    Printf (" The factorial of %d = %d \n", n, fact (n));
}
```

factorial.c

Saved

```c
#include<stdio.h>
int fact(int n)
{
  if(n==0)
    return 1;
  return n*fact(n-1);
}
void main()
{
  int n;
  printf("Enter the value of n:");
  scanf("%d",&n);
  printf("\n%d! = %d",n,fact(n));
}
```

Terminal

```
Enter the value of n:5

5! = 120
Process finished.
```

C-Program: (For Evaluation of Postfix)

1BM19CS090
RAHIL

```
# 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;
```

1BM19CS090
CS090
RAHIL

```
    case '$' :
    case '^' : return pow (op1, op2);
  }
}
void main ()
{ double s[20];
  double res;
  double op1, op2;
  int top, i;
  char postfix [20], symbol;
  pf (" Enter Postfix exprsn: \n");
  sf ("%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 --]
  pf ("result = %f \n", res);
  getch ();
}
```

EvaluationPostfix.c

Saved

```c
#include<stdio.h>
#include<math.h>
#include<string.h>
double eval(char sym,double op1,double op2)
{
  switch(sym)
  {
    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],res,op1,op2;
  int top,i;
  char pf[20],sym;
  printf("Enter the Postfix Expression:");
  scanf("%s",pf);
  top=-1;
  for(i=0;i<strlen(pf);i++)
  {
    sym=pf[i];
    if(isdigit(sym))
      s[++top]=sym-'0';
    else
    {
      op2=s[top--];
      op1=s[top--];
      res=eval(sym,op1,op2);
      s[++top]=res;
    }
  }
  res=s[top--];
  printf("\nEvaluated Result=%f",res);
}
```

```
Terminal

Enter the Postfix Expression:123+*321-+*

Evaluated Result=20.000000
```

prefix.c

Saved

```c
#include<stdio.h>
#include<string.h>
int f(char sym)
{
  switch(sym)
  {
    case '+':
    case '-':return 2;
    case '*':
    case '/':return 5;
    case '#':return -1;
    default:return 6;
  }
}
int g(char sym)
{
  switch(sym)
  {
    case '+':
    case '-':return 3;
    case '*':
    case '/':return 4;
    default:return 7;
  }
}
void rev(char a[30]) {

 int i, j;
 char temp[100];
 for (i = strlen(a)-1,j=0;i+1!=0;--i,++j)
 {
  temp[j] = a[i];
 }
 temp[j] = '\0';
 strcpy(a, temp);
}

void infix_prefix(char infix[],char prefix[])
{
  int top,i,j;
  char s[30],sym;
  top=-1;
  s[++top]='#';
  j=0;
  rev(infix);
  for(i=0;i<strlen(infix);i++)
  {
    sym=infix[i];
    while(f(s[top])>g(sym))
    {
      prefix[j]=s[top--];
      j++;
    }
    if(f(s[top])!=g(sym))
    s[++top]=sym;
    else
    top--;
  }
  while(s[top]!='#')
  prefix[j++]=s[top--];
  prefix[j]='\0';
  rev(prefix);
}
void main()
{
  char infix[20];
  char prefix[20];
  printf("Enter the Valid Infix Expression\n");
  scanf("%s",infix);
  infix_prefix(infix,prefix);
  printf("The Prefix Expression is ");
  printf("\n%s\n",prefix);
}
```

```
Terminal

Enter the Valid Infix Expression
a+b*c+d
The Prefix Expression is
++a*bcd
```

Terminal

```
Enter the Valid Infix Expression
a/b-c/d
The Prefix Expression is
-/ab/cd

Process finished.
```

```
Terminal

Enter the Valid Infix Expression
a+b
The Prefix Expression is
+ab
```