Date 7/10/2020
Page

2) WAP to demonsrrate evaluation of postfix exp.

```
#include <stdio.h>
#include <math.h>
#inuude <sring.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);
  }
 }
 int main ()
 {
  double s[20];
  double res ;
  double op1, op2;
  int top i;
  char postfix[20], symbol;
 printf ("enter postfix exp :\n");
```

Scanned with CamScanner

Date
Page

```
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 is %.f\n", res);
  return 0;
  }
```

OUTPUT -

```
enter postfix exp:
123+* 321-+*
result is 20.000000
```

3) WAP to perform factorial of a number using Recursion

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

OUTPUT -

```
Enter the value of n
5
The factorial of 5 =120
```

4) WAP to perform GCD of 2 numbers using Recursion

```
#include <stdio.h>
int hcf (int n1, int n2);
 int main () {
 int n1, n2;
 printf ("Enter two positive integers : ");
 scanf ("%d %d", &n1, &n2);
 printf ("G.C.D of %d and %d is %d.", n1, n2, hcf (n1, n2));
return 0;
 }
 int hcf (int n1, n2) {
 if (n2! =0)
 return hcf (n2, n1 % n2);
 else
 return n1;
  }
```

OUTPUT—

```
Enter two positive integers : 366 70
G.C.D of 366 and 70 is 2.
Enter two positive integes : 366 60
G.C.D of 366 and 70 is 6.
```

Date
Page

1) WAP to convert infix to prefix expression :

```
#include <stdio.h>
#include <string.h>
#define MAX 20
void infixtoprefix (char infix[20], char prefix[20]);
void reverse (char array[30]);
char pop();
void push (char symbol);
int top = -1;
char stack [MAX];
Main ()
 char infix [20], prefix[20], temp;
 printf ("Enter infix operation:");
 gets (infix);
 infixtoprefix (infix, prefix);
 reverse (prefix);
  puts (prefix);
 }
void infixtoprefix (char infix [20], char prefix [20]) {
 int i, j=0;
 char symbol;
 stack [++top] = '#';
 reverse(infix);
```

Date
Page

```
for (i=0 ; i< strlen (infix); i++) {
  symbol = infix [i];
if (isoperator (symbol) = = 0) {
prefix [j] = symbol;
  j++;
  }
else {
if (symbol = = ')') {
 push (symbol);
 }
else if (symbol = = '(') {
while (stack[top] ! = ')') {
prefix [j] = pop();
 j++;
 }
pop();
}
else {
if (pred (stack[top] < = pred (symbol)) {
push (symbol);
}
else {
while (pred [stack [top]) > = pred (symbol)) {
prefix [j] = pop ();
```

Scanned with CamScanner

Page

```
j++;
}
push (symbol);
  }
 }
}
}
while (stack [top] != '#') {
prefix [j] = pop();
j++;
 }
prefix [j] = '\0';
}
void reverse (char array [30]) {
int i, j;
char temp [100];
for (i = strlen(array) -1, j = 0; i+1 != 0; ++j) {
temp [j] = array[i];
 }
 temp = '\0';
 strcpy (array, temp);
 }
 char pop () {
 char a;
```

Scanned with CamScanner

```
a = stack[top];
top--;
return a;
 }
void push (char symbol) {
top++;
stack [top] = symbol;
  }
int pred (char symbol) {
   switch (symbol) {
case '+' :
case '-' : return 2; break;
 case '*' :
case '/' : return 4; break;
case '$' :
case '^' : return 6; break;
case '#' :
case '(' :
case ')' : return 1; break;
 }
}
int isoperator (char symbol) {
switch (symbol) {
case '+' :
```

```
case '-' :
case '*' :
case '/' :
case '^' :
case '$' :
case '&' :
case '(' :
case ')' : return 1; break;
default : return 0;
    }
  }
```

OUTPUT -

```
Enter infix operation : (a+(b-c)*d)
 +a*-bcd
```