Recursion Functions :-

Write a recursive function to find the factorial of a given number..

```
int fact(int n)
{
   static int f=1;
   if(n)
      {
         f*=n;
         fact(n-1);
      }
   return f;
}
```

Write a recursive function to print the 'n' fibonacciseries numbers.

```
void fibo(int n)
{
   static int i=1,j=1;
   if(n)
   {
      printf("%d  ",i);
      int k=i;
      i=j;
      j=k+i;
      fibo(n-1);
   }
}
```

Write a recursive function to find the sum of digits of a given number.

```
int sod(int n)
{
   static int s=0;

   if(n)
   {
      s=s+n%10;
      sod(n/10);

   }
   return s;
}
```

Write a recursive function to revese the given number.

```
int rev(int n)
{
   static int r=0;

   if(n)
   {
      r=r*10+n%10;
      rev(n/10);
```

```
    }
    return r;
}
```

Write a recursive function to that displays all the proper divisors of a given number exept that and returns their sum.
Ex: 1,3,5,9,15& 45 are the proper divisors of 45.
sum = 1+3+5+9+15
= 33

```
int pod(int n)
{
    static int s=0,i=1;
    if(i<n)
    {
        if(n%i==0)
            s+=i;
        i++;
        pod(n);
    }
    return s;
}
```

Write a recursive function that displays a positive integer in words. For ex: if the integer is 3412 then it is displayed as three four one two.

```
void charnum(int n)
{
    static char *k[]={"zero","one","two","three","four","five","six","seven","eight","nine"};
    if(n)
    {
        charnum(n/10);
        printf("%s ",k[n%10]);

    }
}
```

Write a recursive function to print the palindrome numbers in a given numbers.

```
int palin(int n)
{
    static int k=0;
    if(n)
    {
        k=10*k+n%10;
        palin(n/10);
    }
    if (n==k)
        return 1;
    else
        return 0;


}
```

A number is perfect if the sum of all its positive proper divisors is equal to the number. For example 28 is a perfect number since 28 = 1+2+4+14. Write a recursive function that finds whether a number is perfect or not.

```
int perfect(int n)
{
   static int i=1,s=0;
   if(i<n)
   {
      if(n%i==0)
         s+=i;
      ++i;
      perfect(n);
   }
   if(s==n)
   return 1;
   else
   return 0;
}
```

Write a recursive function to find the largest element in a given Unsorted array.

```
int lar(int *p,int m)
{
   static int l=0,f=0,i=0;
   if(f==0)
   {
      l=p[0];
      f=1;
   }
   if(i<m)
      {
      if(p[i]>l)
         l=p[i];
      ++i;
      lar(p,m);
      }
   return l;
}
```

Write a recursive function to reverse the bits of a given number.

```
int revbit(int n)
{
   static int i=31,j=0,n1=0,f=0;
   if(f==0)
      {
         n1=n;
         f=1;
      }
   if(i>j)
   {
      if(((n1>>i) &1) != ((n1>>j) &1))
         {
```

```
            n1=n1^(1<<i);
            n1=n1^(1<<j);
        }

      i--;
      j++;
      revbit(n1);
   }
   return n1;
}
```

Write a recursive function to revese the elements of a given array.

```
void revarr(int *p,int l)
{
 static int i=0;
 if(i<l-i-1)
 {
    p[i]=p[l-i-1]+p[i]-(p[l-i-1]=p[i]);
    i++;
    revarr(p,l);
}
}
```

Write a recursive function to revese the string. (Note : not just reverse printing charecterby charecter)

```
void revstr(char* p)
{
   static int i=0,j=0,f=0;
   if(f==0)
   {
      j=strlen(p)-1;
      f=1;
   }

   if(i<j)
   {
      p[i]=p[j]+p[i]-(p[j]=p[i]);
      i++;
      j--;
      revstr(p);
   }
}
```

STRCHR fun implementation

```
char* u_strchr(char *p,char a)
{
   static int i=0,f=0;
   if(p[i])
```

```
    {
        if(p[i]==a)
        {
            f=1;
            return &p[i];
        }
        i++;
        u_strchr(p,a);
    }
    if(f==0)
    return 0;
}
```

STRRCHR fun implementation

```
char* u_strrchr(char* p,char c)
{
    static int i=0,f=0;
    static char* t=0;
    if(p[i])
        {
            if(p[i]==c)
                {t=&p[i];f=1;}
            ++i;
        u_strrchr(p,c);
        }
    if(f==0)
        return 0;
    else
        return t;
}
```

STRCAT fun implementation

```
char* u_strcat(char *s1,char *s2)
{
    static int i=0,j=0,f=0;
    if(f==0)
    {
        i=strlen(s1);
        f=1;
    }
    if(s2[j])
    {
        s1[i]=s2[j];
        i++;
        j++;
    u_strcat(s1,s2);
    }
if(s2[j]==0)
    s1[i]='\0';
return s1;
}
```

STRNCAT fun implementation

```
char* u_strncat(char *s1,char *s2,char n)
{
   static int i=0,j=0,f=0;
   if(f==0)
   {
      i=strlen(s1);
      f=1;
   }
   if(j<n)
   {
      s1[i]=s2[j];
      i++;
      j++;
      u_strncat(s1,s2,n);
   }
   if(f==1)
      s1[i]='\0';

   return s1;
}
```

STRCMP fun implementation

```
int u_strcmp(char *s1,char *s2)
{
   static int f=0;
   static char *p1=0,*p2=0;
   if(f==0)
   {
      p1=s1;p2=s2;f++;
   }

   if(*p1 && *p2)
   {
      if(*p1==*p2)
         u_strcmp(++p1,++p2);

   }

   if(*p1==*p2)
      return 0;
   else if(*p1>*p2)
      return 1;
   else
      return -1;
}
```