***1、 编写一个求 n!的函数，然后编写主函数求 e 的值。e=1+1/1!+1/2!+…+…**

```
#include<stdio.h>
float fac(int n)            //求 n!  返回类型是 float  或 double，如果是 int，可能越界
{   int i;
    float p=1;
    for(i=1;i<=n;i++)
        p*=i;
    return p;
}
main()
{   int i=1;
    float e=1;
    do
    {   e+=1/fac(i);            //调用自定义函数 fac，fac(i)就是求 i!
        i++;
    }while(1/fac(i)>=1e-6);     //计算精确到小数点后 6 位
    printf("%f\n",e);
}
```

***2、 编写一个判断一整数是否为素数的函数，然后编写主函数输出 100 以内素数。**

```
#include<stdio.h>
int isprime(int x)
{   int i;
    for(i=2;i<x;i++)
     if(x%i==0)
        return 0;     //如果能被某一数整除，就不是素数，返回 0
    return x;       /*执行到这条语句，一定是做完了 for 循环.而且没有被 i(2~i-1)
                      整除，就是素数，返回这个素数*/
}
main()
{   int i,n=0;
    for(i=3;i<=100;i++)
    {   if( isprime(i)!=0 )
        {   n++;                        //统计素数的个数
            printf("%5d   ",i);
            if(n%5==0)              //每行输出 5 个素数
                printf("\n");
        }
    }
}
```

*3、 编写一个求 $x^n$ 的函数，然后编写主函数求 $1+2+2^2+2^3+\ldots+2^{63}$ 的值。

根据数学知识，我们知道这个值是 $2^{64}-1=18446744073709551615$。但在 VC 中 unigned long int 最大也只能表示到 $2^{32}-1$。那么我们选用 double 来实现。程序如下：

```
#include<stdio.h>
double power(int m,int n)  //求 m^n
{  int i;
   double p=1;
   for(i=1;i<=n;i++)
       p*=m;
   return p;
}
main()
{  int i;
   double sum=1;
   for(i=1;i<63;i++)
       sum+=power(2,i);
   printf("%lf\n",sum);
}
```

结果：18446744073709552000.000000

这个结果并不精确，因为 double 的数据只有 17 位有效数字位。

**4、 编写一个函数，将一个十进制数转换为二进制形式输出。

```
#include<stdio.h>
void dectobin(int x)          //递归
{   if(x/2>0)
       dectobin(x/2);
    printf("%d",x%2);
}
main()
{   int x;
    printf("请输入要转换的十进制数");
    scanf("%d",&x);
    dectobin(x);
}
```

为了同学们便于理解递归，按递归思想用非递归形式写出程序如下：该程序是以五位的二进制数为例。

```
#include<stdio.h>
void dectobin(int i)
{   if (i>2)
    {     if((i/2)>2)
          {    if((i/2)/2>0)
               {    if(   ((i/2)/2)/2   >0)
                    {    if( (((i/2)/2)/2)/2 > 0)
                         printf("%d",(((i/2)/2)/2)/2%2);
                    }
                    printf("%d",((i/2)/2)/2 %2);
               }
               printf("%d",(i/2)/2%2);
```

```
            }
            printf("%d",(i/2)%2);
    }
    printf("%d",i%2);
}
```

**5、用二分法求方程 $x^2-3=0$ 的根。

```
#include<stdio.h>
#include<math.h>
double f(double x)  {   return x*x-3;}
//注意，方程定义成一个函数，这样解方程的函数就可以与具体方程无关。
double root(double a,double b)
{   double x;
    x=(a+b)/2;
    while(fabs(f(x))>1e-6)
    {   if(f(x)*f(a)>0)   //同号
            a=x;
        else                //异号
            b=x;
         x=(a+b)/2;
    }
    return x;
}
main()
{   double x=root(0,5);
    printf("f(%lf)=%lf\n",x,f(x));
}
```

***6、用二分法求方程 $x^2-3=0$ 的根，要求用递归方法来实现。

```
#include<stdio.h>
#include<math.h>
double f(double x){   return x*x-3;}
double root(double a,double b)
{   double x = (a+b)/2;
    if(fabs(f(x))<1e-6)
        return x;
    else
        if(f(x)>0)
            return root(a,x);
        else
            return root(x,b);
}
main()
{   double x=root(0,5);
    printf("f(%lf)=%lf\n",x,f(x));
}
```

**7、 编写一个函数用辗转法求二个数的最大公约数。然后编写一个主函数，求二个整数的最大公约数。

```
#include<stdio.h>
int gcd(int m,int n)
{    int t,r;
     if(m<n)
      t=m,m=n,n=t;
     while(m%n)
     {      r=m%n;
            m=n;
            n=r;
     }
     return n;
}
main()
{    int x,y;
     scanf("%d%d",&x,&y);
     printf("%d\n",gcd(x,y));
}
```

***8、 编写一个函数用辗转法求二个数的最大公约数。要求用递归实现。然后编写一个主函数，求二个整数的最大公约数。

```
#include<stdio.h>
int gcd(int m,int n)
{    int t;
     if(m<n)
            t=m,m=n,n=t;
     if(m%n)
            return gcd(n,m%n);
     else
            return n;
}
main()
{    int x,y;
     scanf("%d%d",&x,&y);
     printf("%d\n",gcd(x,y));
}
```