```c
//calcula el numero combinatorio
//n sobre m
#include<stdio.h>
int main()
{
  //fn factorial de n
  //fm factorial de m
  //fnm factorial de n-m
  //c numero combinatorio
  int n,m,fn,fm,fnm,i,c;
  printf("\nIntroduce n y m:");
  scanf("%d%d",&n,&m);
  //calculo del factorial de n
  fn=1;
  for (i=2;i<=n;i++)
    fn=fn*i;
  //calculo del factorial de m
  fm=1;
  for (i=2;i<=m;i++)
    fm=fm*i;
  //calculo del factorial de n-m
  fnm=1;
  for (i=2;i<=n-m;i++)
    fnm=fnm*i;
  //calculo del numero combinatorio
  c=fn/fm/fnm;
  printf("\n(%d,%d)=%d",n,m,c);
}
```

```c
//calcula el numero combinatorio
//n sobre m con funciones
#include<stdio.h>
//devuelve el factorial de n
int factorial(int n)
{
    int f=1,i;
    for (i=2;i<=n;i++)
       f=f*i;
    return f;
}

//devuelve el numero combinatorio n sobre m
int combina(int n, int m)
{
    return factorial(n)/factorial(m)/factorial(n-m);
}

int main()
{
    int n,m;
    printf("\nIntroduce n y m:");
    scanf("%d%d",&n,&m);
    printf("\n%d sobre %d=%d",n,m,combina(n,m));
}
```

```c
//calcula e elevado a x
#include<stdio.h>
float ex(float x, float precision, int maxiter)
{
    int n=0;
    float e_x=0,t_n=1,t_n_1;
    //mientras no se alcance la precision
    //y mientras no se superen el maximo
    //numero de iteraciones
    while(t_n>precision&&n<maxiter)
        {
        //suma del termino actual
        e_x=e_x+t_n;
        //actualizacion de n
        n++;
        //actualizacion del termino anterior
        t_n_1=t_n;
        //calculo del siguiente en base al anterior
        t_n=t_n_1*x/n;
        }
    return e_x;
}

int main()
{
float x,p;
int iteraciones;
printf("\nIntroduce x, precision, iteraciones:");
scanf("%f%f%d",&x,&p,&iteraciones);
printf("\ne elevado a %f, con precision %f e iteraciones %d=%f",
       x,p,iteraciones,ex(x,p,iteraciones));
}
```

```c
//n primeros capicuas
#include<stdio.h>
//devuelve un numero con las cifras
//en orden inverso
int reves(int n)
{
    //inicialmente el numero del reves es cero
    int r=0;
    //mientras queden cifras en el original
    while(n!=0)
        {
        //muevo (multiplico por 10) las cifras
        //que hay en el numero del reves a la izquierda
        //y añado la nueva cifra
        r=r*10+n%10;
        //siguiente cifra
        n=n/10;
        }
    return r;
}
//devuelve 1 si es capicua
//cero en caso contrario
int capicua(int n)
{
    int es_capicua;
    //si el numero es igual que el mismo al reves
    if (n==reves(n))
      //es capicua = 1
      es_capicua=1;
    else
      es_capicua=0;
    return es_capicua;
}

int main()
{
    int n,conta=0,num=0;
    printf("\nCuantos:");
    scanf("%d",&n);
    //mientras no se contabilicen n capicuas
    while(conta<n)
       {
           //Si el numero actual es capicua
           if(capicua(num))
             {
                 //lo muestro
                 printf("\n%d",num);
                 //lo cuento
                 conta++;
             }
           //siguiente numero
           num++;
       }

}
```

```c
//esto es un ejemplo de paso
//por referencia, con printf dentro de
//las funciones, para seguir el programa
//y ver como cambian las variables
#include<stdio.h>
//esta funcion esta mal. Como se pasan
//los parametros por valor, se intercambian
//las copias, no las variables de la llamada
void intercambia_mal(int a, int b)
{
  int tmp;
  printf("\nDentro de intercambia_mal:");
  printf("\nValores antes de intercambiar %d %d",a,b);
  tmp=a;
  a=b;
  b=tmp;
  printf("\nDentro de intercambia_mal:");
  printf("\nValores despues de intercambiar %d %d",a,b);
  printf("\nSalgo de la funcion");
}

//esta esta bien, se usa paso por referencia
//se intercambian los valores de las variables
//de la llamada
void intercambia_bien(int *pa, int *pb)
{
  int tmp;
  printf("\nDentro de intercambia_bien:");
  printf("\nValores antes de intercambiar %d %d",*pa,*pb);
  tmp=*pa;
  *pa=*pb;
  *pb=tmp;
  printf("\nDentro de intercambia_bien:");
  printf("\nValores despues de intercambiar %d %d",*pa,*pb);
  printf("\nSalgo de la funcion");
}

int main()
{
  int x=1,y=2;
  printf("\nAntes de llamar a intercambia_mal");
  printf("\nValores %d %d",x,y);
  //llamada a intercambia_mal
  intercambia_mal(x,y);
  printf("\nDespues de llamar a intercambia_mal");
  printf("\nValores %d %d",x,y);
  //llamada a intercambia_bien, notar los &
  //&x es un puntero a x, &y es un puntero a y
  intercambia_bien(&x,&y);
  printf("\nDespues de llamar a intercambia_bien");
  printf("\nValores %d %d",x,y);
}
```

```
//salida del programa
// Antes de llamar a intercambia_mal
// Valores 1 2
// Dentro de intercambia_mal:
// Valores antes de intercambiar 1 2
// Dentro de intercambia_mal:
// Valores despues de intercambiar 2 1
// Salgo de la funcion
// Despues de llamar a intercambia_mal
// Valores 1 2
// Dentro de intercambia_bien:
// Valores antes de intercambiar 1 2
// Dentro de intercambia_bien:
// Valores despues de intercambiar 2 1
// Salgo de la funcion
// Despues de llamar a intercambia_bien
// Valores 2 1
```

```
//e elevado a x con parametro adicional para informar
//sobre la condicion de terminacion
#include<stdio.h>
float ex(float x, float precision, int maxiter, int *estado
)
{
    int n=0;
    float e_x=0,t_n=1,t_n_1;
    //mientras no se alcance la precision
    //y mientras no se superen el maximo
    //numero de iteraciones
    while(t_n>precision&&n<maxiter)
        {
        //suma del termino actual
        e_x=e_x+t_n;
        //actualizacion de n
        n++;
        //actualizacion del termino anterior
        t_n_1=t_n;
        //calculo del siguiente en base al anterior
        t_n=t_n_1*x/n;
        }
    //si t_n es <= precision, ha parado
    //por esa condicion
    if (t_n<=precision)
        *estado=1;
    //en caso contrario por alcanzar
    //el maximo numero de iteraciones
    //considero que no se van a dar las
    //dos condiciones simultaneamente
    else
        *estado=0;
    return e_x;
}
```

```
int main()
{
float x,p;
int iteraciones,condicion_salida;
printf("\nIntroduce x, precision, iteraciones:");
scanf("%f%f%d",&x,&p,&iteraciones);
printf("\ne elevado a %f, con precision %f e iteraciones %d=%f",
       x,p,iteraciones,ex(x,p,iteraciones,&condicion_salida
));
if (condicion_salida)
   printf("\nHa parado por precision");
else
   printf("\nHa parado por iteraciones");
}

//ejemplos de ejecucion
//para por iteraciones
// Introduce x, precision, iteraciones:2 .000001 4
// e elevado a 2.000000, con precision 0.000001 e iteracion
es 4=6.333333
// Ha parado por iteraciones
//para por precision
// Introduce x, precision, iteraciones:2 .001 100
// e elevado a 2.000000, con precision 0.001000 e iteracion
es 100=7.388713
// Ha parado por precision
```

```
#include<stdio.h>
//separa un float en parte entera y decimal
//ejemplo en el que se producen dos resultados
//no pueden devolverse con un unico return
//se pueden devolver en dos parametros por referencia
void separa(float n, float *pentera,float *pdecimal)
{
  //asigno n como entero a *pentera, se pierden los
  //decimales
  *pentera=(int)n;
  //la parte decimal es el original
  //menos la parte entera
  *pdecimal=n-*pentera;
}

int main()
{
  float a=7.3,b,c;
  separa(a,&b,&c);
  printf("%f %f %f",a,b,c);
}
//salida del programa
//7.300000 7.000000 0.300000
```

```c
//escribir una funcion que calcule
//la media de los n primeros numeros
//mayores o iguales que 1
//tales que la suma de la primera
//de sus cifras con la ultima
//es divisible entre una cifra recibida
//como parametro y que además no
//contienen a ese digito
#include<stdio.h>

//devuelve la primera cifra
int primera(int n)
{
  return n%10;
}

//devuelve la ultima cifra
int ultima(int n)
{
  int cifra=0;
  while(n!=0)
    {
      cifra=n%10;
      n=n/10;
    }
  return cifra;
}

//devuelva la suma de la
//primera y la ultima
int suma_primera_ultima(int n)
{
  return primera(n)+ultima(n);
}

//devuelve 1 si n contiene a cifra
//0 en caso contrario
int contienex(int n,int cifra)
{
  while(n!=0)
    {
      if (n%10==cifra)
        return 1;
      n=n/10;
    }
  return 0;
}
```

```c
//devuelve la media de los n
//primeros numeros tales que
//...
float  media_rara(int n,int c)
{
  int conta=0,num=1;
  float suma=0;
  while(conta<n)
    {
      if(suma_primera_ultima(num)%c==0&&
         !contienex(num,c))
        {
          conta++;
          suma=suma+num;
        }
      num++;
    }
  return suma/n;
}
//devuelve una cifra != 0
int pide_cifra_no_cero(void)
{
  int cifra;
  do{
    printf("\nIntroduce una cifra:");
    scanf("%d",&cifra);
  }while(cifra<=0||cifra>9);
  return cifra;
}
//devuelve una cifra > 0
int pide_mayor_cero(void)
{
  int num;
  do{
    printf("\nIntroduce un numero mayor que cero:");
    scanf("%d",&num);
  }while(num<=0);
  return num;
}
int main()
{
  int cantidad,cifra;
  float m;
  cantidad=pide_mayor_cero();
  cifra=pide_cifra_no_cero();
  m=media_rara(cantidad,cifra);
  printf("\n%f",m);
}
```

```c
//mostrar los n primeros numeros
//tales que la suma de la mayor
//de sus cifras con la menor de
//sus cifras sea divisible por
//un digito mayor que cero
//introducido por el teclado
#include<stdio.h>

//devuelve la mayor de las
//cifras de n
int mayor(int n)
{
  int cifra=n%10,mayor=cifra;
  n=n/10;
  while(n!=0)
    {
      cifra=n%10;
      if (cifra>mayor)
        mayor=cifra;
      n=n/10;
    }
  return mayor;
}

//devuelve la menor de las
//cifras de n
int menor(int n)
{
  int cifra=n%10,menor=cifra;
  n=n/10;
  while(n!=0)
    {
      cifra=n%10;
      if (cifra<menor)
        menor=cifra;
      n=n/10;
    }
  return menor;
}

//devuelve la suma de la
//mayor y de la menor de las
//cifras
int suma_mayor_menor(int n)
{
  return mayor(n)+menor(n);
}

//devuelve una cifra distinta
//de cero
int pide_cifra_no_cero(void)
{
  int cifra;
  do{
    printf("\nIntroduce una cifra:");
    scanf("%d",&cifra);
  }while(cifra<=0||cifra>9);
  return cifra;
}

//devuelve un numero mayor
//que cero
int pide_mayor_cero(void)
{
  int num;
  do{
    printf("\nIntroduce un numero mayor que cero:");
    scanf("%d",&num);
  }while(num<=0);
  return num;
}

//en el main el esquema
//comentado en clase
int main()
{
  int num=0,n,conta=0,c;
  n=pide_mayor_cero();
  c=pide_cifra_no_cero();
  //mientras no encuentre n
  //numeros que cumplan la
  //condicion
  while(conta<n)
    {
      //si se cumple la condicion
      if(suma_mayor_menor(num)%c==0)
        {
          //lo cuento
          conta++;
          //lo muestro
          printf("\n%d",num);
        }
      //siguiente numero
      num++;
    }
}
```