```
class Euclid(
     list,
     Euclid_Utils,
     Euclid_Tests
 ):

 ##!
 ##! Euclid Class creator.
 ##!

 def __init__(x,y=0,b=10):
     x.sign=1

     if (type(y)==type(x)):
         x.b=y.b
     else:
         x.b=b

     if (isinstance(y,int)):
         if (y>0):
             s=str(y)
             x.Calloc(len(s))
             for i in range(len(s)):
                 x[i]=int(s[i])


             x.reverse()
             #print("==",x)
     elif (isinstance(y,list) or isinstance(y,Euclid)):
         x.Calloc(len(y))
         for i in range( len(y) ):
             x[i]=int(y[i])
 ##!
 ##! ....
 ##!

 ##!
 ##! Floor division of two Euclids: overload  // operator.
 ##!


 def __floordiv__(x,y):
     if (y>x or x.b!=y.b):
         print(
             "Euclid: Division Error",y,">",x
         )
         exit()

     xx=x.Calc10()
     yy=y.Calc10()

     c=xx//yy

     c=Base10To(c,x.b)

     return c.Trim_Digits()
```

```
    ##!
    ##! Modulo of two Euclids: overload  % operator..
    ##!

    def __mod__(x,y):
        if (y>x or x.b!=y.b):
            print(
                "Euclid: Division Error",y,">",x
            )
            exit()

        xx=x.Calc10()
        yy=y.Calc10()

        r=xx%yy

        r=Base10To(r,x.b)

        return r.Trim_Digits()

    ##!
    ##! Reverses an Euclid, returning the reversed one.
    ##!

    def Reverse(x):
        y=Euclid(b=x.b)
        for i in range(len(x)-1,-1,-1):
            y.append(x[i])

        return y


    ##!
    ##! Test reverse divisibility on Euclid x.
    ##!

    def Reverse_Test(x):
        res=False
        y=x.Reverse()
        if (x>y):
            c=x//y
            r=x%y

            if (r.Calc10()==0 and c.Calc10()>1):
                res=True

        return res
##!
##! From the list of Euclids, xs, return the ones that pass Reverse_Test.
##!

def Euclids_Reverse_Test(xs):
    xres=[]
    for x in xs:
        if (x.Reverse_Test()):
            xres.append(x)
```

```
    print("\t",len(xs),"Euclids generated")

    return xres

##!
##! Generate list of numbers in base b of exactly n digits.
##! If given, append(!) lasts as digits )(augmenting number of digits).
##!

def NDigits_List(b,ndigits,lasts=[]):
    ciphers=[]
    numbers=[]
    for i in range(b):
        ciphers.append(i)

        #Avoid numbers with leading 0s.
        if (i>0):
            numbers.append([i])

    for n in range(ndigits):
        rnumbers=[]
        for lst in numbers:
            for cipher in ciphers:
                cp_lst=list(lst)
                cp_lst.append(cipher)
                rnumbers.append(cp_lst)

        numbers=rnumbers

    for i in range(len(numbers)):
        #list concatenation (NOT __add__!!
        numbers[i]=Euclid(numbers[i]+lasts,b)

    return numbers

#Main execution

#Base and number of digits.
b=12
N=4

All_Reverse_Test(b,ndigits)
```