```
>
>
> 2^256
```

```
115792089237316195423570985008687907853269984665640564039457584007913129639936\
    6                                                                          (1)
```

```
> text := "streams of words are flowing past like endless rain into a paper cup";
        text := "streams of words are flowing past like endless rain into a paper cup"   (2)
>
```

given upper bound on n, find primes p and q so that n=p*q is no bigger than that.

```
> rand( );
                                  395718860534                                    (3)
> bigr := rand(10^20..10^21) :
> randomize( ); bigr( );
                                  1397571881
                                308134388516675346170                             (4)
> maxn := 10^20;
                          maxn := 100000000000000000000                           (5)
> bigr := rand(floor(sqrt(maxn/10))..floor(sqrt(maxn))) :
> bigr( );
                                  8649161650                                      (6)
> p := nextprime(bigr( ));
  q := nextprime(bigr( ));
                                  p := 5388787739
                                  q := 8312389079                                 (7)
> n := p·q;
                          n := 44793700350712702381                               (8)
> phi := (p − 1)·(q − 1);
                          ϕ := 44793700337011525564                               (9)
> maxn − n;
                                 55206299649287297619                             (10)
> ifactor(phi);
                  (2)^2 (7) (17) (19) (83) (158493757) (376501)                   (11)
> e := rand( );
  while (gcd(e, phi) > 1) do
    e := rand( );
  od;
                                  e := 61653250038
                                 e := 191623515867
                                 e := 477167989563                                (12)
> d := 1/e mod phi;
                          d := 34991633761399256787                               (13)
> SetupRSA:=proc(maxn::posint)
```

```
    local p,q,e,d,phi,bigr,n;
    bigr := rand(floor(sqrt((1/10)*maxn)) .. floor(sqrt(maxn)));
    p := nextprime(bigr());
    q := nextprime(bigr());
    n:=p*q;
    phi:=(p-1)*(q-1);
    e := rand(); while gcd(e, phi) > 1 do e := rand() end do;
    d := 1/e mod phi;
    return( [n,e], [n,d]);
  end:
> public,private:=SetupRSA(10^30);
```

public, *private* := [498934970132026922784382701893, 357649609075], [498934970132026922784382701893, 294062316400037353676096194891] (14)

```
> private;
```

[498934970132026922784382701893, 294062316400037353676096194891] (15)

```
> text;
```

"streams of words are flowing past like endless rain into a paper cup" (16)

```
> nums := convert(text, bytes); # analog of StringToList
```

nums := [115, 116, 114, 101, 97, 109, 115, 32, 111, 102, 32, 119, 111, 114, 100, 115, 32, 97, 114, 101, 32, 102, 108, 111, 119, 105, 110, 103, 32, 112, 97, 115, 116, 32, 108, 105, 107, 101, 32, 101, 110, 100, 108, 101, 115, 115, 32, 114, 97, 105, 110, 32, 105, 110, 116, 111, 32, 97, 32, 112, 97, 112, 101, 114, 32, 99, 117, 112] (17)

```
> floor(log[128](n)); # how big a k-graph we want
```

9 (18)

```
> kgraphs := convert(nums, base, 128, 128^9); #$ this is StringToKgraph
```

kgraphs := [8016916883531676275, 7008110262347485286, 7985972299952567026, 7626990723245281383, 8343959714394551019, 7985589634731905139, 7341297860210079732, 30311764082] (19)

```
> # encode via RSA:
> n:=public[1];e:=public[2]; d:=private[2];
```

$n := 498934970132026922784382701893$

$e := 357649609075$

$d := 294062316400037353676096194891$ (20)

```
> map(x->x&^e mod n, kgraphs);
```

[313749736436878942295924428987, 307587739090828399235254356788, 228793301308026686791144331 83, 381602433372854461631097117947, 267869933664649383890721172821, 351588142907695847676777142420, 328110288123260426156192510 12, 284403681975418055367167852532] (21)

```
> map(x→x&^d mod n, %);
```

[8016916883531676275, 7008110262347485286, 7985972299952567026, 7626990723245281383, 8343959714394551019, 7985589634731905139, 7341297860210079732, 30311764082] (22)

```
> EncodeMsg:=proc(text::string, k::posint)
    convert(convert(text,bytes), base, 128, 128^k);
  end:
  DecodeMsg:=proc(nums::list, k::posint)
    convert(convert(nums, base, 128^k, 128), bytes);
```

```
end:
> EncodeMsg(text, 9);
[8016916883531676275, 7008110262347485286, 7985972299952567026,
  7626990723245281383, 8343959714394551019, 7985589634731905139,
  7341297860210079732, 30311764082]                                    (23)
> DecodeMsg(%, 9);
        "streams of words are flowing past like endless rain into a paper cup"   (24)
>
```