1
```
restart;maple_mode(1);cas_setup(0,0,0,1,0,1e-10,10,[1,50,0,25],0,0,0); #radians,pas de cmplx, pas de  Sqrt
```
Menu

2
```
l:={1,4,5,6,7};a:=rand(10):
```
(1 4 5 6 7 , Done)
Menu

3
```
member(a,l);
```
0
Menu

4
```
a; #verification.
```
9
Menu

5
```
7 + 4 mod 3; (7+4)mod 3;
```
(8, 2)
Menu

6
```
maple_mode(1); # en mode maple
```
Warning: some commands like subs might change arguments order
Menu

7
```
ln(exp(1));ln(e);  #ca n'est que la lettre e, pas le reel.
```
(1, ln(e))
Menu

8
```
maple_mode(0);
```
Warning: some commands like subs might change arguments order
Menu

9
```
evalf(e); //la doc fait reference au mode xcas.
```
2.718281828
Menu

10
```
maple_mode(1);
```
Warning: some commands like subs might change arguments order
Menu

11
```
evalf(exp(1));
```
2.718281828
Menu

12
```
ln(exp(1));
```
1
Menu

13
```
log(exp(1)); # les 2 marchent
```
1
Menu

14
```
bete comme le crible d'eratostene sont en O(p), et bete n'use pas de memoire.
donc asymptotiquement, le crible n'est pas avantageux.
```

15 Prog Edit Add | nxt | OK | Save

```
bete:=proc(n)
i:=3;
a:=0;
SQ:=evalf(sqrt(n)); #  pour ne le faire qu'une fois. ne  pas le mettre dans le  whi
while (i<SQ)
   do if (n mod i) <>0  then i:=i+2;
      else a:=i;i:=n;
      fi ;
   od;
if a<>0 then a; else n fi;
end;
```

```
// Warning: i a SQ  declared as global variable(s)
// End defining bete
```

```
proc(n)
 i:=3;
 a:=0;
 SQ:=evalf(sqrt(n));
 while i<SQ do
  if (irem(n,i))<>0 then
    i:=i+2 else
    a:=i;
    i:=n
  fi ;
 od;;
 if a<>0 then
  a else
  n
 fi ;

end;
```
Menu

16 i:=4;N:=nextprime(10^i+5432)*nextprime(2*10^i+1234);

(4, 328032433) Menu

17 bete(N);

15439 Menu

18 i:=5;N:=nextprime(10^i+5432)*nextprime(2*10^i+1234); #i=6 passe encore mais...

(5, 21218879939) Menu

19 bete(N);

Evaluation time: 0.7

105437 Menu

20 Attention, pour que la suite r'ecurrente soit bien programmee, il ne faut pas recalculer tous les termes jusque u_i ni u_2i a chaque etape!

21 Prog Edit Add | nxt | OK | Save

```
pollard :=  proc(n)
local x,y ;
x:=1; y:=x ;
while member ( igcd(y-x,n) ,   {1,n} ) do
     x:=(x^2+1) mod n ;
     y:=((y^2+1)^2+1) mod n ;
      od ;
igcd(y-x,n);
end  proc;
```

// Success
// End defining pollard

```
proc(n)
  local x,y;
  x:=1;
  y:=x;
  while member(igcd(y-x,n),{1,n}) do
    x:=irem(x^2+1,n);
    y:=irem((y^2+1)^2+1,n);
   od;;
  igcd(y-x,n);

end;
```

Menu

22 N:=nextprime(10^6)*nextprime(2*10^6):

Done Menu

23 pollard(N);

1000003 Menu

24 N:=nextprime(10^8+5*rand(1000))*nextprime(2*10^8+11*rand(1000)):

Done Menu

25 pollard(N);

Evaluation time: 0.57

100003753 Menu

26 pollard est en O(sqrt(p)) ssi la fonction suivante est bornee.

27 Prog Edit Add | nxt | OK | Save

```
comptep:= proc(n)
local x,y,j ;
x:=1; y:=x ;j:=0;
while member ( igcd(y-x,n) ,   {1,n} ) do
     x:=(x^2+1) mod n ;
     y:=((y^2+1)^2+1) mod n ;
     j:=j+1; od ;
evalf(j/(sqrt(igcd(y-x,n))));
```

```
// Success
// End defining comptep
```

```
proc(n)
 local x,y,j;
 x:=1;
 y:=x;
 j:=0;
 while member(igcd(y-x,n),{1,n}) do
  x:=irem(x^2+1,n);
  y:=irem((y^2+1)^2+1,n);
  j:=j+1;
 od;;
 evalf(j/(sqrt(igcd(y-x,n))));

end;
```

Menu

28 On cr'ee une liste de valeurs. Il ne faut pas des nombres trop petits pour que pollard ait de bonne chance d'aboutir. On fait imprimer a chaque etape pour voir s'il blocque a une etape.

29
```
l:=[];
for i from 7 to 30 do
N:=nextprime((rand(3^(i))))*nextprime((rand(2^(i+1))));
t:=comptep(N);
print(i,t);
l:=append(l,t);
N:=nextprime((rand(3^(i))))*nextprime((rand(2^(i+1))));
t:=comptep(N);
print(i,t);
l:=append(l,t);
N:=nextprime((rand(3^(i))))*nextprime((rand(2^(i+1))));
t:=comptep(N);
print(i,t);
l:=append(l,t);
```

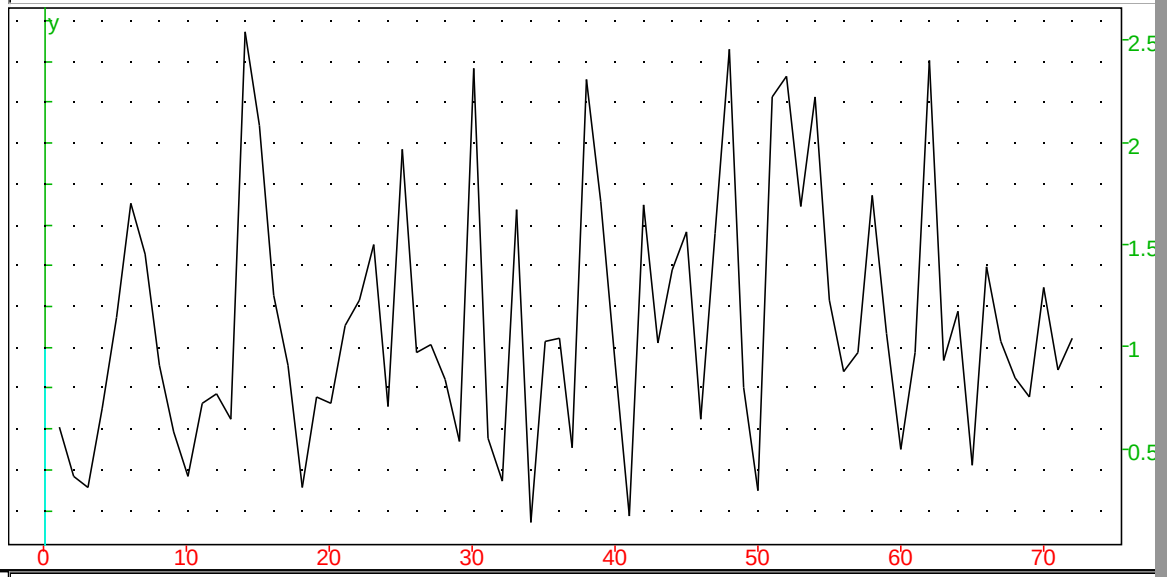
```
10,0.729805165425
10,0.77586019249
11,0.651024795545
11,2.5408117487
11,2.08813066644
12,1.25108648434
12,0.914566141275
12,0.317097551741
13,0.755635701501
13,0.724584757579
13,1.10426810153
14,1.22750796879
14,1.50038041779
14,0.707452731553
15,1.9712088492
15,0.973895037141
15,1.01481827343
16,0.840939777479
16,0.539203099529
16,2.36640645005
17,0.551761725234
17,0.342321996755
17,1.6724354808
18,0.146689760419
18,1.03296731032
18,1.04093892927
19,0.510611504311
19,2.31094197589
19,1.71042450778
20,0.931511110213
20,0.176860132028
```

```
21,1.02053300715
21,1.38061829304
21,1.56378505004
22,0.650208434654
22,1.55418114676
22,2.4598044248
23,0.804627119976
23,0.301114827339
23,2.22537205888
24,2.32803073406
24,1.69198294425
24,2.22596233945
25,1.2314962783
25,0.882834342695
25,0.97170508154
26,1.74409042202
26,1.06718019563
26,0.50363557224
27,0.976842868841
27,2.40148644007
27,0.936531471363
28,1.17413510179
28,0.421549435318
28,1.3929171267
29,1.02606013551
29,0.853800109133
29,0.758136072031
30,1.29227333363
30,0.892445360811
30,1.04846151603
Evaluation time: 13.95
```

[0.6115928397 0.370891428 0.31330418 0.7022468832 1.145457504 1.701143931 1.455468278]

Menu

30 `plotlist(l); #ca a bien l'air borne`



31 `P:=i->product(1-j/p,j=1..i-1);`

```
// Warning: j p  declared as global variable(s)
// End defining P
```

$$i \to \prod_{j=1}^{i-1}\left(1-\frac{j}{p}\right)$$

Menu

32 `i:=2;limit(log(P(i))/(i*(i-1)/2/p),p,+infinity);`

(2, -1)

Menu

33 On devine que P(i) equivaut a: -i*(i-1)/(2p), on l'illustre ainsi:

34 `l:=limit(log(P(2))/(i*(i-1)/2/p),p,+infinity);`

35 for i from 3 to 50 do l:=l,limit(log(P(i))/(i*(i-1)/2/p),p,+infinity) od;

-1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 ,

Menu

36 Puisque ln est croissante, on se demande quand ln(P(i))>ln(0.5). On compare
donc -ln(0.5) avec i^2/2p

37 sqrt(-2*ln(0.5)); #ca fait a peu pres le 1.18 de l'enonce

1.177410023

Menu

38 u_2i=u_i est mauvais pour cette suite recurent, c'est en O(p) comme la methode bete.
En effet u_2i=a^i u_i +c(a^i-1)/(a-1) [p], donc u_2i=u_i ssi
(a^i-1)(-ui+c/(a-1))=0[p]. Mais a^i=1[p] par ex si a generateur,
alors i>=p-1

39

undef

Menu

40 illustration: on fait afficher le rapport entre le nombre de tours et sqrt(p).

41 a:=54321123;c:=nextprime(10^4);

(54321123 , 10007)

Menu

42

43 Prog Edit Add | nxt | OK | Save

```
comptelin:= proc(n)
local x,y ;
x:=1; y:=x ;j:=0;
while member ( igcd(y-x,n) ,   {1,n} ) do
     x:=(a*x+c) mod n ;
     y:=(a*(a*y+c)+c) mod n ;
     j:=j+1; od ;
evalf(j/sqrt(igcd(y-x,n)));
end proc;
```

// Warning: j a c declared as global variable(s)
// End defining comptelin

```
proc(n)
  local x,y;
  x:=1;
  y:=x;
  j:=0;
  while member(igcd(y-x,n),{1,n}) do
    x:=irem(a*x+c,n);
    y:=irem(a*(a*y+c)+c,n);
    j:=j+1;
   od;;
  evalf(j/(sqrt(igcd(y-x,n))));

end;
```

Menu

44 l:=[];rand(3^4);

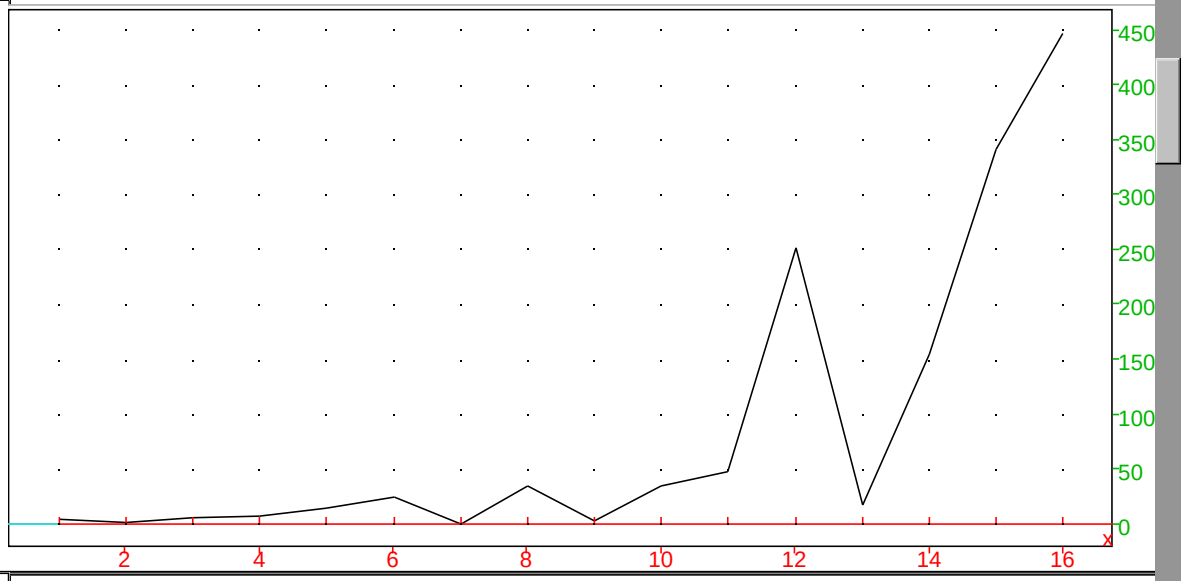
```
45 for i from 4 to 19 do
 N:=nextprime(rand(3^i))*nextprime(rand(2^(i+1)));
 t:=comptelin(N);
print(i,t);
l:=append(l,t);
od;
```

```
5,1.37479175006
6,5.26803686295
7,7.98442062786
8,14.86621957
9,25.2784801771
10,0.608857765456
11,34.0734626923
12,3.35489480445
13,34.4564950716
14,48.3967976903
15,251.922607195
16,18.0084857202
17,155.545009571
18,341.31180759
19,447.210241391
Evaluation time: 19.15
```

```
3.880570001  1.37479175  5.268036863  7.984420628  14.86621957  25.27848018  0.6088577655  34.07
```

Menu

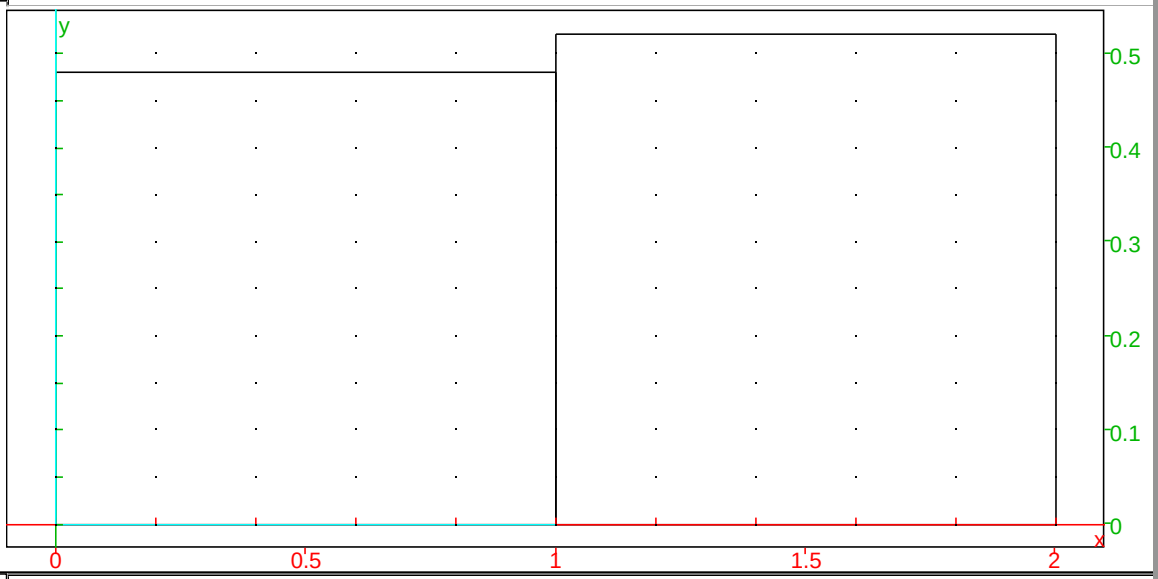
```
46 plotlist(l); # ca n'a plus l'air borne
```




```
47 l:=[seq(rand(2),i=1..100)]:
```

Done

```
48 histogram(classes(l,0,1)); #On commence a 0, largeur constante 1.
```

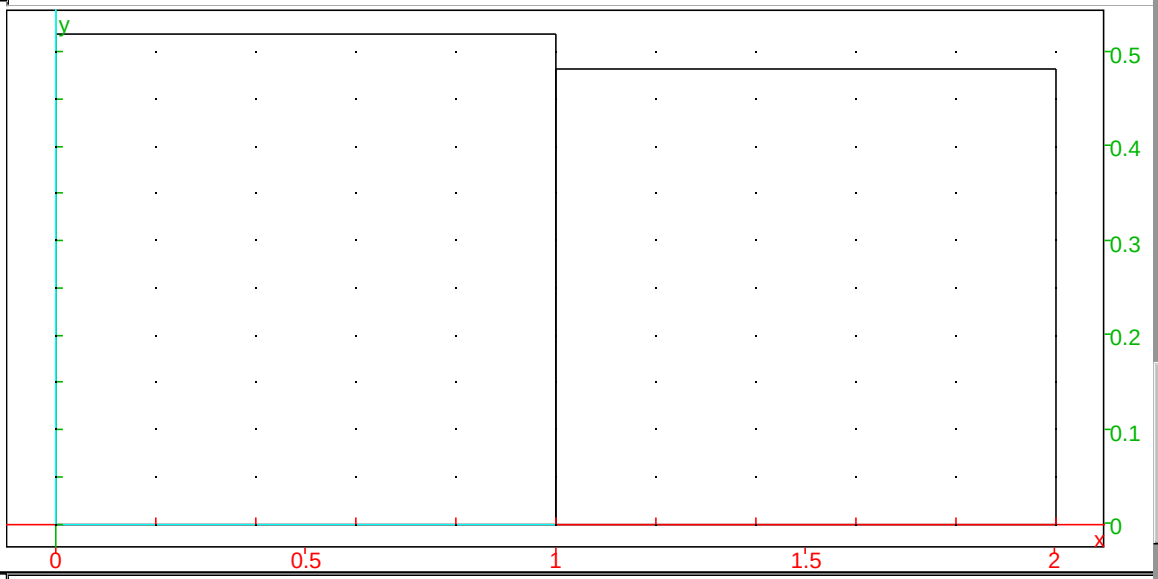



```
49 l:=[seq(rand(2),i=1..1000)]:
```

Done

Menu

```
50 histogram(classes(l,0,1));
```




```
51 N:=nextprime(10^6)*nextprime(2*10^6);
```

2000009000009

Menu

```
52 u:=n->if n=0 then 2 else ((u(n-1))^2+1) mod N fi;
// Warning: u N  declared as global variable(s)
// End defining u
u: recursive definition
```

```
if n=0 then
  2 else
  irem((u(n-1))^2+1,N)
n   ->  fi
```

Menu

```
53 u(10); # u(10000);Error, (in u) too many levels of recursion
```

1997309223146

Menu

```
54
```

55
```
etudesuite := proc(n,M)
local x,l;
x:=12345;l:=[];
for i from 1 to M do
    x:=(x^2+1) mod n ;
    l:=[op(l),x];
    od ;
l;
end:
```

```
// Warning: i  declared as global variable(s)
// End defining etudesuite
```

```
proc(n,M)
  local x,l;
  x:=12345;
  l:=[];
  for i from 1 to M+1/2 do
  x:=irem(x^2+1,n);
  l:=[op(l),x];
   od;;
  l;

end;
```

Menu

56
```
donnees:=(etudesuite(N,2000));
```

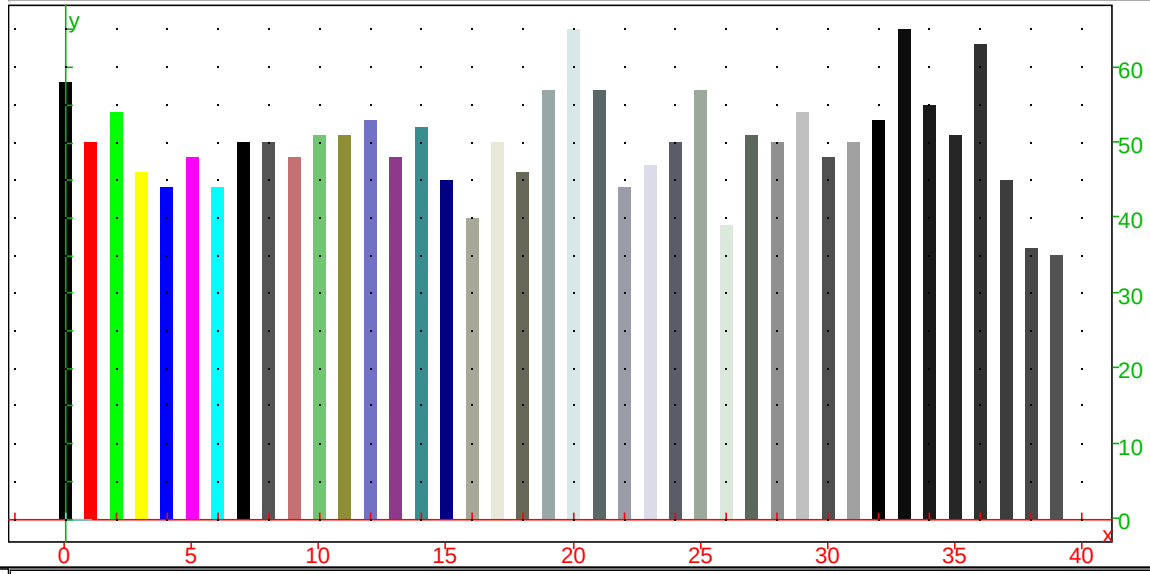
```
152399026   1358617644169   1831711515025   190274858284   1218880810174   893513062655   5230676
```

Menu

57
```
cldonnees:=classes(donnees,0,N/40); #40 classes
```

0.0 .. 5.0000225e+10	58
5.0000225e+10 .. 1.0000045e+11	50
1.0000045e+11 .. 1.50000675e+11	54
1.50000675e+11 .. 2.000009e+11	46
2.000009e+11 .. 2.50001125e+11	44
2.50001125e+11 .. 3.0000135e+11	48
3.0000135e+11 .. 3.50001575e+11	44
3.50001575e+11 .. 4.000018e+11	50
4.000018e+11 .. 4.50002025e+11	50
4.50002025e+11 .. 5.0000225e+11	48
5.0000225e+11 .. 5.50002475e+11	51
5.50002475e+11 .. 6.000027e+11	51
6.000027e+11 .. 6.50002925e+11	53
6.50002925e+11 .. 7.0000315e+11	48
7.0000315e+11 .. 7.50003375e+11	52
7.50003375e+11 .. 8.000036e+11	45
8.000036e+11 .. 8.50003825e+11	40
8.50003825e+11 .. 9.0000405e+11	50
9.0000405e+11 .. 9.50004275e+11	46
9.50004275e+11 .. 1.0000045e+12	57
1.0000045e+12 .. 1.050004725e+12	65
1.050004725e+12 .. 1.10000495e+12	57
1.10000495e+12 .. 1.150005175e+12	44
1.150005175e+12 .. 1.2000054e+12	47

58 diagramme_batons(cldonnees); #on peut cacher les noms dans le menu Cfg



59 ----------Test d'une variante de pollard----------
cette variante de pollard pour minimiser le nombre de igcd n'apporte pas d'amalioration
sous xcas, elle me semble meme un peu plus longue!!!!!!!

60 Prog Edit Add | nxt | OK | Save

```
pollard2 :=  proc(n)
local x,y,c,,yy,xx,pp ;
xx:=1; yy:=xx ;pp:=1;
while member(igcd(pp,n),  {1}) do
x:=xx;y:=yy;
for c from 1 to 20 do
     xx:=(xx^2+1) mod n ;
     yy:=((yy^2+1)^2+1) mod n ;
pp:=(pp*(xx-yy)) mod n;
od ;
od;
while member ( igcd(y-x,n) ,  {1} ) do
     x:=(x^2+1) mod n ;
     y:=((y^2+1)^2+1) mod n ;
      od ;
     pp:=igcd(y-x,n);
if (pp<>n)  then pp else print( "pas trouve" ) fi;
end proc:
```

// Success
// End defining pollard2

Done Menu

61 N:=nextprime(10^6)*nextprime(2*10^6):

Done Menu

62 pollard2(N);

1000003 Menu

63 N:=nextprime(10^8+5*rand(1000))*nextprime(2*10^8+11*rand(1000)):

Done Menu

64 pollard2(N);

100002937 Menu

65 pollard(N);

100002937 Menu

66