Μικρή εισαγωγή στο gp-pari.

Στο σύστημα μας δίνουμε την εντολή

```
$ gp
```

και το σύστημα ανταποκρίνεται με

```
GP/PARI CALCULATOR Version 2.1.4 (released)
                i686 running linux (ix86 kernel) 32-bit version
                (readline v4.1 enabled, extended help available)

                       Copyright (C) 2002 The PARI Group

PARI/GP is free software, covered by the GNU General Public License, and
comes WITHOUT ANY WARRANTY WHATSOEVER.

Type ? for help, \q to quit.
Type ?12 for how to get moral (and possibly technical) support.

   realprecision = 28 significant digits
   seriesprecision = 16 significant terms
   format = g0.28

parisize = 4000000, primelimit = 500000
?
```

και είναι έτοιμο να δεχτεί εντολές. Στο ερωτηματικό δίνουμε για παράδειγμα την εντολή

```
?Pi
```

οπότε πέρνουμε ως απάντηση το

```
%1 = 3.141592653589793238462643383
?
```

Μπορούμε να ορίσουμε με πόσα δεκαδικά σημαντικά ψηφία θέλουμε να γίνουν οι υπολογισμοί. Έτσι δίνουμε

```
? \p 1500
   realprecision = 1502 significant digits (1500 digits displayed)
? Pi
```

```
%2 = 3.14159265358979323846264338327950288419716939937510582097494459230781640
628620899862803482534211706798214808651328230664709384460955058223172535940
812848111745028410270193852110555964462294895493038196442881097566593344612847
5648233786783165271201909145648566923460348610454326648213393607260249141273
7245870066063155881748815209209628292540917153643678925903600113305305488204
6652138414695194151160943305727036575959195309218611738193261179310511854807
4462379962749567351885752724891227938183011949129833673362440656643086021394
9463952247371907021798609437027705392171762931767523846748184676694051320005
6812714526356082778577134275778960917363717872146844090122495343014654958537
1050792279689258923542019956112129021960864034418159813629774771309960518707
2113499999983729780499510597317328160963185950244594553469083026425223082533
4468503526193118817101000313783875288658753320838142061717766914730359825349
0428755468731159562863882353787593751957781857780532171226806613001927876611
1959092164201989380952572010654858632788659361533818279682303019520353018529
6899577362259941389124972177528347913151557485724245415069595082953311686172
7855889075098381754637464939319255060400927701671139009848824012858361603563
7076601047101819429555961989467678374494482553797747268471040475346462080466
8425906949129331367702898915210475216205696602405803815019351125338243003558
7640247496473263914199272604269922796782354781636009341721641219924586315030
2861829745557067498385054945885869269956909272107975093029
?
```

Το pari ξέρει και από συναρτήσεις, έτσι ας δώσουμε

```
? (x^3-1)/(x-1)
%3 = x^2 + x + 1
```

ή ας υπολογίσουμε την σειρά Taylor του $sin(x)$.

```
taylor(sin(x),x)
%4 = x - 1/6*x^3 + 1/120*x^5 - 1/5040*x^7 + 1/362880*x^9 - 1/39916800*x^11
+ 1/6227020800*x^13 - 1/1307674368000*x^15 + O(x^16)
```

Αν δώσουμε

```
? 2/6
%5 = 1/3
```

Παρατηρήστε ότι στο τελευταίο αποτέλεσμα απλοποιήθηκε το κλάσμα, αλλά δεν υπολόγισε την δεκαδική παράσταση του ενός τρίτου. Αυτό είναι χαρακτηριστικό των προγραμμάτων που κάνουν πράξεις *συμβολικά* σε αντίθεση με τους υπολογισμούς κινητής υποδιαστολής. Για να δούμε τα πλεονεκτήματα των ακριβών υπολογισμών ας υπολογίσουμε την ορίζουσα ενός πίνακα του Hilbert με ακριβή τρόπο και με υπολογισμό κινητής υποδιαστολής

```
? \p 8
realprecision = 8 significant digits
? a=mathilbert(20);
? matdet(a)*1.
%6 = 4.2061789 E-226
? matdet(a*1.)
%7 = 2.3821891 E-147
```

Τα αποτελέσματα δεν έχουν καμία σχέση! Όπως θα μάθετε σε ένα μάθημα Αριθμητικής Ανάλυσης, οι πίνακες του Hilbert έχουν *κακή κατάσταση* και οι πράξεις με αυτούς δεν δίνουν αξιόπιστα αποτελέσματα σε υπολογισμούς κινητής υποδιαστολής.

Ας κάνουμε έναν υπολογισμό ανάλυσης σε πρώτους παράγοντες

```
? #
   timer = 1 (on)
? factor(1234567234987234982374982374923847923847329847239 8)
time = 450 ms.
%8 =
[2 1]

[124769 1]

[49040951 1]

[24614186399893 1]

[409858256465941175163 97 1]
```

Ο ειδικός χαρακτήρας ♯ που δόθηκε στην πρώτη γραμμή έχει σαν αποτέλεσμα το πρόγραμμα να μετράει τον χρόνο που έκανε για να πραγματοποιήσει ένα υπολογισμό. Το αποτέλεσμα της παραγοντοποίησης του 12345672349872349823749823749238479238473298472398 διαβάζεται ως εξή-ς: Ο υπολογισμός έγινε σε $450ms$ και ο αριθμός γράφεται σαν $2 \cdot 49040951 \cdot 24614186399893 \cdot 409858256465941175163 97$.

Να δούμε ένα ποιο σύνθετο παράδειγμα το οποίο είναι ένα προγραμματάκι που υπολογίζει τους πρώτους αριθμούς του Mersenne, δηλαδή τους πρώτους αριθμούς της μορφής $2^n - 1$ για $n = 1, 200$. (ας σημειωθεί ότι ο μεγαλύτερος από αυτούς δηλαδή ο $2^{200} - 1$ είναι αριθμός με 60 δεκαδικά ψηφία.

```
? for(x=1,200,if(isprime(2^x-1),print(2^x-1)))
```

```
3
7
31
127
8191
131071
524287
2147483647
2305843009213693951
618970019642690137449562111
162259276829213363391578010288127
170141183460469231731687303715884105727
```

Από ότι βλέπετε δεν υπάρχουν και πολλοί πρώτοι του Mersenne Η λογική του προγράμματος αυτού είναι να διατρέξουμε όλους τους αριθμούς από το 1 μέχρι το 200 και να τυπώσουμε στην οθόνη μόνο τους πρώτους.

Το pari έχει ενσωματωμένο σύστημα βοήθειας. Έτσι δίνοντας ένα ερωτηματικό σαν εντολή μας εμφανίζει μια λίστα με τις συναρτήσεις που καταλαβαίνει, ενώ με ερωτηματικό ακολουθούμενο από μία εντολή μας δίνει την ακριβή της σύνταξη

```
? ?factor
factor(x,{lim}): factorization of x. lim is optional and can be set
whenever x is of (possibly recursive) rational type. If lim is set
return partial factorization, using primes up to lim (up to primelimit
if lim=0).
```