CHAPTER 9
Primes

A *prime* number is an integer (grater than 1) that is not divisible by any positive integer other than 1 and itself (of course, by divisibility we mean exact divisibility). The numbers 2, 3, 5, 7 and 11 are prime, but 4, 6, 8, 9 and 10 are not. Non-prime numbers are called *composite*.

There is no general formula for prime numbers, but Euclid showed (in about 300 B.C.) that there are infinitely many primes. We also know that primes occur les frequently among large numbers.

The testing of primes is of considerable interest. Attention has arisen recently because of cryptography.

A simple and straightforward method of determining if a number N is prime is called the Sieve of Erastosthenes. (Erastisthenes of Cyrene was a Greek mathematician, 276-196 B.C., who also calculated the circumference of the Earth.) The idea is to write down all the integers from 1 to N. Then leave 2 and strike out all even numbers after 2. The next number after 2 which has not been struck out is prime. This is 3. Now strike out every third number after 3. The next number left after 3 is 5 which must be prime. Now strike out every fifth number after 5. This process is continued. What remains are the primes between 1 and N.

The table below shows the results of a sieve on the numbers up to 100. The multiples of 2 are crossed out by /, the multiples of 3 by -, 5 by / and 7 by |.

	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	49	~~50~~
~~51~~	~~52~~	53	~~54~~	~~55~~	~~56~~	~~57~~	~~58~~	59	~~60~~
61	~~62~~	~~63~~	~~64~~	~~65~~	~~66~~	67	~~68~~	~~69~~	~~70~~
71	~~72~~	73	~~74~~	~~75~~	~~76~~	77	~~78~~	79	~~80~~
~~81~~	~~82~~	83	~~84~~	~~85~~	~~86~~	~~87~~	~~88~~	89	~~90~~
~~91~~	~~92~~	~~93~~	~~94~~	~~95~~	~~96~~	97	~~98~~	~~99~~	~~100~~

The following program uses the Sieve of Erastosthenes to produce a list of prime numbers less than some given number N. This number is INPUT at the start.

Listing 9.1
LIST

```
   10 REM Sieve of Erastosthenes
   20 MODE 6:PRINT ' TAB(9);"Sieve of Er
astosthenes"':@%=10
   30 PRINT "This program will calculate
 the prime   numbers up to some given nu
mber."'
   40 REPEAT
   50  INPUT '"Up to what number: ";N
   60  IF N<4 OR INT(N)<>N OR N>5000 THE
N PRINT '"Be reasonable!"
   70 UNTIL N>3 AND INT(N)=N AND N<5001
   80 DIM A%(N):CLS:PRINT '"Primes from
2 to ";N
   90 FOR I=2 TO N
  100  IF A%(I)=0 THEN PROCPrime
  110 NEXT
  120 PRINT CHR$(7) ''' TAB(10);"Another
 go? Y or N ";
  130 REPEAT:G$=GET$:UNTIL G$="Y" OR G$=
"N"
  140 IF G$="Y" THEN RUN
  150 CLS:PRINT '"Bye for now.":END
  160 DEF PROCPrime
  170 IF 39-POS<LEN(STR$(I)) THEN PRINT
  180 PRINT ;I;" ";
  190 FOR J=I TO N STEP I:A%(J)=I:NEXT
  200 ENDPROC
```

RUN

```
          Sieve of Erastosthenes

This program will calculate the prime
numbers up to some given number.
```

```
Up to what number: ?100


Primes from 2 to 100

2  3  5  7  11  13  17  19  23  29  31  37  41  43
47  53  59  61  67  71  73  79  83  89  97



          Another go? Y or N
```

Note that for large numbers the program starts off slowly; but soon starts printing primes very fast. Printing the primes between 2 and 5000 takes about 43 seconds on the BBC micro and about 59 seconds on the Electron.

The Sieve of Erastosthenes is comceptually easy. It is useful if you want a list of prime numbers. But it is not a very practical method of testing whether a number is prime.

A simple way to check if a number N is prime is to check whether it is divisible by the numbers smaller than N, step by step. The following illustrates this method. A clock is included to indicate how long it takes to test the number for primality.

Listing 9.2
LIST

```
   10 REM Primes Version 1
   20 MODE 1:COLOUR 3:PRINT '"   Ineffic
ient Prime tester Version 1"':@%=10:COLO
UR 1
   30 REPEAT
   40  INPUT '"Number to be tested: ";N%
   50  IF N%<4 THEN COLOUR 3:PRINT '"Be
reasonable!":COLOUR 1
   60 UNTIL N%>3
   70 COLOUR 3:PRINT '"The number being
tested is ";N%
   80 TIME=0:A$="":X%=N%-1
   90 FOR I%=2 TO X%
```

```
  100  IF N% MOD I% = 0 THEN A$="not ":I
%=X%
  110 NEXT
  120 COLOUR 2:PRINT '"The number is ";A
$;"prime."
  130  PRINT '"Time taken to test number
 ";INT(TIME/100+0.5);" seconds."
  140 COLOUR 3:PRINT CHR$(7) '' TAB(10);
"Another go? Y or N ";
  150 REPEAT:G$=GET$:UNTIL G$="Y" OR G$=
"N"
  160 IF G$="Y" THEN RUN
  170 CLS:PRINT '"Bye for now.":END
```

RUN

```
   Inefficient Prime tester Version 1


Number to be tested: ?9001

The number being tested is 9001

The number is prime.

Time taken to test number 20 seconds.


           Another go? Y or N
```

The program works but it really is inefficient and slow. For instance to test the primality of 9001 takes 20 seconds on the BBC micro and about 56 seconds on the Electron. To test the primality of a larger number such as 90001 takes about 205 seconds on the BBC and about 590 on the Electron. A little thought will produce enormous benefits.

For a start we needn't bother using all the numbers between 2 and N. We need only use the numbers between 2 and INT(SQR(N)). Because if M is an integer which divides N and is greater than INT(SQR(N)) then N/M

is an integer that divides N and is smaller than INT(SQR(N)). This simple addition is included in version two below.

Listing 9.3
LIST

```
   10 REM Primes Version 2
   20 MODE 1:COLOUR 3:PRINT '"   Ineffic
ient Prime tester Version 2"':@%=10:COLO
UR 1
   30 REPEAT
   40  INPUT '"Number to be tested: ";N%
   50  IF N%<4 THEN COLOUR 3:PRINT '"Be
reasonable!":COLOUR 1
   60 UNTIL N%>3
   70 COLOUR 3:PRINT '"The number being
tested is ";N%
   80 TIME=0:A$="":X%=SQR(N%)
   90 FOR I%=2 TO X%
  100  IF N% MOD I% = 0 THEN A$="not ":I
%=X%
  110 NEXT
  120 COLOUR 2:PRINT '"The number is ";A
$;"prime."
  130  PRINT '"Time taken to test number
 ";INT(TIME/100+0.5);" seconds."
  140 COLOUR 3:PRINT CHR$(7) '' TAB(10);
"Another go? Y or N ";
  150 REPEAT:G$=GET$:UNTIL G$="Y" OR G$=
"N"
  160 IF G$="Y" THEN RUN
  170 CLS:PRINT '"Bye for now.":END
```

RUN

```
   Inefficient Prime tester Version 2


Number to be tested: ?90001

The number being tested is 90001

The number is prime.

Time taken to test number 1 seconds.


           Another go? Y or N



   Inefficient Prime tester Version 2

Number to be tested: ?9001

The number being tested is 9001

The number is prime.

Time taken to test number 0 seconds.


           Another go? Y or N
```

This version works considerably faster. Testing the number 9001 now takes just over 1 second. A larger number, such as 987654323 takes about 87 seconds on the BBC and about 252 seconds on the Electron. Don't attempt to test such a large number with version 1 - unless you enjoy looking mindlessly at your television screen.

Remark. The addition incorporated in Prime Version 2 could also be

incorporated in the program Erastosthenes to produce Erastosthenes 2. The result would improve the speed.

We can further improve the program Primes Version 2 by taking a tip from the Sieve of Erastosthenes. We can miss out all even numbers bigger than 2 and miss out every third number after 3. These suggestions have been added to the third version of the program.

Listing 9.4
LIST

```
   10 REM Primes Version 3
   20 MODE 1:COLOUR 3:PRINT ' TAB(9);"Pr
ime tester Version 3"':@%=10:COLOUR 1
   30 REPEAT
   40  INPUT '"Number to be tested: ";N%
   50  IF N%<8 THEN COLOUR 3:PRINT '"Be
reasonable!":COLOUR 1
   60 UNTIL N%>7
   70 COLOUR 3:PRINT '"The number being
tested is ";N%
   80 TIME=0:A$="":X%=SQR(N%)
   90 IF N% MOD 2 = 0 OR N% MOD 3 = 0 TH
EN A$="not " ELSE PROCOther
  100 COLOUR 2:PRINT '"The number is ";A
$;"prime."
  110 PRINT '"Time taken to test number
";INT(TIME/100+0.5);" seconds."
  120 COLOUR 3:PRINT CHR$(7) '' TAB(10);
"Another go? Y or N ";
  130 REPEAT:G$=GET$:UNTIL G$="Y" OR G$=
"N"
  140 IF G$="Y" THEN RUN
  150 CLS:PRINT '"Bye for now.":END
  160 DEF PROCOther
  170 FOR I%=5 TO X% STEP 6
  180 IF N% MOD I% = 0 THEN A$="not ":I%
=X%
```

```
  190 IF N% MOD (I%+2) = 0 THEN A$="not
":I%=X%
  200 NEXT
  210 ENDPROC
```

RUN

```
          Prime tester Version 3


Number to be tested: ?90001

The number being tested is 90001

The number is prime.

Time taken to test number 0 seconds.


            Another go? Y or N
```

This speeds up the program a little. Testing the prime 987654323 now takes about 30 seconds on the BBC and about 86 seconds on the Electron. The largest integer that your computer can store is 2147483647, is it a prime number?

A composite number may be written as a product of prime numbers. For instance

6 = 3 * 2
24 = 3 * 2 * 2 * 2
81018001 = 9001 * 9001

and so on. The various primes that occur are called *factors* of the number. By adding a few extra lines to version 3 we can have all the factors of a number displayed.

The next program prints out the factors of a number.

Listing 9.5
LIST

```
   10 REM Prime factors
   20 MODE 1:COLOUR 3:PRINT ' TAB(13);"P
rime factors"':@%=10:COLOUR 1
   30 REPEAT
   40  INPUT '"Number to be tested: ";N%
   50  IF N%<8 THEN COLOUR 3:PRINT '"Be
reasonable!":COLOUR 1
   60 UNTIL N%>7
   70 COLOUR 3:PRINT '"The factors of ";
N%;" are:"':COLOUR 2
   80 TIME=0:A$="":X%=SQR(N%):S%=X%+1
   90 IF N% MOD 2 = 0 THEN PROCFactor(2)
  100 IF N% MOD 3 = 0 THEN PROCFactor(3)
  110 FOR I%=5 TO X% STEP 6
  120  IF N% MOD I% = 0 THEN PROCFactor(
I%)
  130  IF N% MOD (I%+2) = 0 THEN PROCFac
tor(I%+2)
  140  IF I%>S% THEN I%=X%
  150 NEXT
  160 IF 39-POS<LEN(STR$(N%)) THEN PRINT
  170 IF N%>1 THEN PRINT ;N%
  180 COLOUR 1:PRINT '"Time to find fac
tors: ";INT(TIME/100+0.5);" seconds."
  190 COLOUR 3:PRINT CHR$(7) '' TAB(10);
"Another go? Y or N ";
  200 REPEAT:G$=GET$:UNTIL G$="Y" OR G$=
"N"
  210 IF G$="Y" THEN RUN
  220 CLS:PRINT '"Bye for now.":END
  230 DEF PROCFactor(J%)
  240 REPEAT
  250  IF 39-POS<LEN(STR$(J%)) THEN PRIN
T
  260  PRINT ;J%;" ";:N%=N% DIV J%
  270 UNTIL N% MOD J% > 0
```

```
  280 S%=SQR(N%)+1
  290 ENDPROC
```

RUN

```
              Prime factors


Number to be tested: ?162036002

The factors of 162036002 are:

2 9001 9001

Time to find factors: 9 seconds.

           Another go? Y or N


              Prime factors


Number to be tested: ?1022048

The factors of 2048 are:

2 2 2 2 2 2 2 2 2 2 2

Time to find factors: 0 seconds.

           Another go? Y or N
```

Large primes

Finding large prime numbers is a pastime for some. The largest known primes are usually Mersenne primes. Numbers of the form

$$2^P - 1$$

are called *Mersenne numbers* because of the French monk, Father Marin Mersennes, who made some suggestions concerning the primality of such numbers.

Many early writers believed that the Mersenne numbers are prime if the exponent M is prime. For the first few cases this is indeed true.

$$2^2 - 1 = 3$$
$$2^3 - 1 = 7$$
$$2^5 - 1 = 31$$
$$2^7 - 1 = 127$$

But for P = 11 the number is not prime.

$$2^{11} - 1 = 2047 = 23 * 89$$

For at least the last 100 years the world's largest known prime has always been a Mersenne prime (except for a short period in 1951). Prior to January 1983, the largest known prime was

$$2^{44497} - 1$$

a Mersenne prime discovered by David Slowinski in April 1979. This was the 27th Mersenne prime known. In about 1983 Slowinski found the much larger prime

$$2^{86243} - 1$$

which is a Mersenne prime containing 25962 decimal digits. (It would fill several pages of this book to write out the number completely.) The computer that Slowinski used was a Cray-1. This is an amazingly fast machine, even so it took 1 hour 36 minutes and 22 seconds of computer time to perform the test.

An arbitrary number of the size $2^{286243} - 1$ would be impossible to test for primality using any of the programs in the previous section. But there are special techniques for Mersenne numbers. These were developed during the last 100 years. To test a number of the form $N = 2^P - 1$ for primality one defines a sequence as follows:

$U_1 = 4$
$U_2 = (U_1 * U_1 - 2) \text{ MOD } N$
•
•
•
$U_{p-1} = (U_{p-2} * U_{p-2} - 2) \text{ MOD } N$

Then N is prime if and only if $U_{P-1} = 0$. This means that in order to test for N being prime we need to perform approximately P simple calculations. Performing 86243 operations would be quite simple for your computer if the numbers involved were small. But here one is dealing with numbers that have 86243 binary digits and your computer stores 32 binary digits. There are methods available to get around this problem but we won't go into the details here. See the next chapter.

Probabilistic primality testing

Given unlimited time we could test for the primality of an integer by trial division. But time is limited, even for a computer.

In the last few years a new test has been devised which is based on an old theorem of Pierre Fermat who lived during the 17th century. Fermat showed that if P is a prime number and B is some other number between 1 and P - 1 then the number $8^{P-1} - 1$ is divisible by P. For instance, let P = 11 and B = 2 then $2^{11-1} - 1$ is 1023 which is indeed divisible by 11.

Although Fermat proved his theorem for all values of B, the Chinese mathematician Pomerance (5th Century B.C.) knew the theorem in the case B = 2. In addition he believed, incorrectly, that the converse is true. In other words he believed that if $2^{P-1} - 1$ is divisible by P then P is a prime number. However the number 341 divides $2^{340} - 1$ and 341 is not prime. We call such a number a *pseudoprime to the base 2*. In general a composite number P that divides $B^{P-1} - 1$ is called a *pseudoprime to the base B*. Most numbers that appear to be pseudoprimes are in fact genuine primes. And this is what the test is based on.

Listing 9.6
LIST

```
   10 REM Probabilisic primality test
   20 MODE 1:COLOUR 3:PRINT ' TAB(6);"Pr
obabilistic primality test"':@%=10:COLOU
R 1
   30 REPEAT
   40  INPUT '"Number to be tested: ";N%
```

```
   50  IF N%<8 THEN COLOUR 3:PRINT '"Be
reasonable!":COLOUR 1
   60 UNTIL N%>7
   70 REPEAT
   80  REM Factorise N%-1 = (2^T%)*X%
   90  T%=0:X%=N%-1
  100  REPEAT
  110   IF X% MOD 2 = 0 THEN T%=T%+1:X%=
X% DIV 2
  120  UNTIL X% MOD 2 = 1
  130  REM Select a base B
  140  I=RND(-TIME):B%=RND(N%-3) + 1
  150  COLOUR 1:PRINT '"Test using base
";B%:COLOUR 2
  160  REM Raise B% to power X%
  170  P%=1
  180  REPEAT
  190   IF X% MOD 2 = 1 THEN P%=P%*B%:P%
=P% MOD N%
  200   B%=B%*B%:B%=B% MOD N%:X%=X% DIV
2
  210  UNTIL X%=0
  220  REM Check B%^X%
  230  REPEAT
  240   TEST=-1
  250   IF P%=1 OR P%=N%-1 THEN TEST=0
  260   IF TEST AND T%<2 THEN PRINT '"Nu
mber is NOT prime.":TEST=1:G$="N"
  270   IF TEST THEN P%=P%*P%:P%=P% MOD
N%:T%=T%-1
  280  UNTIL NOT TEST
  290  IF TEST=0 PRINT '"Number is PROBA
BLY prime."'"Do you want to test the nu
mber again    with another base? Y or N"
'
  300  IF TEST=0:REPEAT:G$=GET$:UNTIL G$
="Y" OR G$="N"
  310 UNTIL G$="N"
```

```
  320 COLOUR 3:PRINT CHR$(7) '' TAB(10);
"Another go? Y or N ";
  330 REPEAT:G$=GET$:UNTIL G$="Y" OR G$=
"N"
  340 IF G$="Y" THEN RUN
  350 CLS:PRINT '"Bye for now.":END
```

RUN

```
      Probabilistic primality test


Number to be tested: ?341

Test using base 29

Number is PROBABLY prime.

Do you want to test the number again
with another base? Y or N


Test using base 31

Number is NOT prime.


          Another go? Y or N
```

Try numbers such as 341 and 561, neither of which is prime. If you find that they are a pseudoprime to some base then try another base.