A Supplement to

An Introduction to The Theory of Numbers

Fifth Edition

by

Ivan Niven, Herbert S. Zuckerman, Hugh L. Montgomery

John Wiley & Sons, Inc.

Publication history:

First Edition	December, 1992
Second Edition	September, 1994

Preface to the Second Edition

Throughout its long history, number theory has been characterized by discovery based upon empirically observed numerical patterns. By using a computer with appropriate software, the student can now inspect data that is both more extensive and more accurate than in former times. With this in mind, a set of 70 programs has been prepared for use in the classroom as an aid to instruction, for use by students in individual study and exploration, and also in structured laboratories. These programs are written in Borland's Turbo Pascal version 7.0, running under DOS on IBM PC-compatible machines. Both the source code and the compiled code are provided; these programs may be freely copied and distributed to students using the text. Some of these programs, such as FacTab and PowerTab, display data in which patterns may be detected. Other programs, such as EuAlDem1 and PwrDem1a, offer demonstration of specific algorithms that are employed in computations. Finally, a third class of programs, typified by Factor and GCD, perform useful calculations on demand. The programs relevant to a particular section are listed in the table Programs by Section. Before embarking on a section, the instructor may wish to experiment with these programs, in order to become familiar with their operation.

It was intended that the algorithms employed in the accompanying programs should be limited to those discussed in the text, so that the student would be in a position to understand exactly what each program is does. As the programs developed, a few exceptions to this rule crept in, as follows: In the program Ind, for calculating the index (i.e., discrete logarithm),of a number modulo p a method of Shanks is used. This is explained in Laboratory 12, in the documentation of the program, and also in the demonstration provided by the program IndDem. In the program ProveP, which is based on Problem 39 at the end of §2.8, an extra device invented by H. C. Williams has been added. For details see the description of this program in the *Reference Guide to Turbo Pascal Programs*, in this manual. The scheme for calculating the Lucas functions, described in §4.4, has not been followed, because the one sidestep formula involves division by 2, which is problematic when the calculations are being done modulo m with m even. For an account of the method actually used, see the description of the program Lucas in the *Reference Guide to Turbo Pascal Programs*. The disadvantage of using only those algorithms found in the text is that in some cases faster execution could have been achieved by using some other algorithm. This particularly the case with programs that involve factoring (the quadratic sieve method is faster), proving primality (the Atkin-Morain and the Adleman-Rumely methods are faster), or locating the roots of polynomial congruences modulo p (the Cantor-Zassenhaus method is *much* faster).

If your students have experience in programming, you may wish to make the source code of these programs available to them. By examining the source code, a student may see in detail how a particular algorithm has been implemented. On the other hand, an effort has been made to design programs whose operation is so natural that very little time will be needed to learn how to use them. Thus students unfamiliar with computers or programming should have no difficulty.

In the accompanying programs, an integer variable n in is usually declared to be of type `integer` if $|n| \le 30{,}000$, or `longint` if $|n| \le 10^9$, or of type `comp` if $|n| \le 10^{18}$. No provisions are made for integers larger than this, a disadvantage in some contexts. The advantage is that one avoids the extra commands and variable types that a multiple precision arithmetic package would use.

Some of the laboratories assume that the student has access not only to the Turbo Pascal programs, but also to UBASIC, a wonderful BASIC interpreter designed by Yuji Kida. For more details see *Further Reseources*.

If you encounter any problem with the operation of the Turbo Pascal programs, or have suggestions for their improvement, please communicate your comments to me at `hlm@math.lsa.umich.edu`, or by snail mail. CLINT users who wish to be kept abreast of future revisions should ask me to add their name to the list of CLINTERS.

These computational laboratories are still in an experimental stage. More labs and programs are needed. In addition, the labs may be too long, or too difficult, or may ask the wrong questions. Any thoughts you have would be appreciated. You may want to compose your own, but it is hoped that the ones here at least offer inspiration. If you want to edit these labs to customize them for your own use, you can obtain the .tex files that create them by ftp–see the *Further Resources* for details.

It is a pleasure to thank A. O. L. Atkin, J. D. Brillhart, H. Flanders, D. E. G. Malm, C. Pomerance, J. L. Selfridge, R. C. Vaughan, and S. S. Wagstaff Jr. for their help with algorithmic and technical issues.

Hugh L. Montgomery
6 September, 1994

Contents

Programs by Type

CALCULATIONS

Carmichael function car(m)	`car` [m]
Chinese Remainder Theorem	`crt` [a_1 m_1 a_2 m_2]
determinant modulo m	`detmodm`
discrete logarithm base g of a modulo p	`ind` [g a p]
factor n	
by trial division	`factor` [n]
by $p-1$ method	`p-1` [n [a]]
by rho method	`rho` [n [c]]
find next prime	`getnextp` [x]
greatest common divisor	`gcd` [b c]
index base g of a modulo p	`ind` [g a p]
Jacobi symbol $\left(\frac{P}{Q}\right)$	`jacobi` [P Q]
Lucas functions U_n, V_n modulo m	`lucas` [n [a b] m]
multiply residue classes modulo m	`mult` [a b m]
order of a modulo m	`order` [a m [c]]
phi function $\phi(n)$	`phi` [n]
$\pi(x)$	`pi` [x]
power a^k modulo m	`power` [a k m]
primitive root of prime p	`primroot` [p [a]]
prove primality of p	`provep` [p]
rational approximation to decimal	`rat` [x]
reduce $ax^2+bxy+cy^2$	`reduce a b c`
represent n as sum of s k-th powers	`sumspwrs` [n s k]
roots of	
$ax \equiv b \pmod{m}$	`lincon` [a b m]
$f(x) \equiv 0 \pmod{p^j}$	`hensel`
$P(x) \equiv 0 \pmod{m}$	`polysolv`
$x^2 \equiv a \pmod{p}$	`sqrt` [a p]
$A\mathbf{x} = \mathbf{b}$ in integers	`simlinde`
square root modulo p	`sqrt` [a p]
strong pseudoprime test of m base a	`spsp` [[a] m]

DEMONSTRATIONS

Chinese Remainder Theorem	`crtdem`
determinants modulo m	`detdem`
discrete logarithm base g of a modulo p	`inddem` [g a p]
Euclidean algorithm	`eualdem1`, `eualdem2`, `eualdem3`

factorization	
by $p-1$ method	`p-1dem`
by rho method	`rhodem` [`n`]
greatest common divisors	`fastgcd`, `slowgcd`
(see also Euclidean algorithm)	
heapsort algorithm	`hsortdem`
index base g of a modulo p	`inddem` [`g a p`]
Jacobi symbol $\left(\frac{P}{Q}\right)$	`jacobdem` [`P Q`]
linear congruence $ax \equiv b \pmod{m}$	`lncndem` [`a b m`]
Lucas functions	`lucasdem` [`n` [`a b`] `m`]
multiplication of residue classes	`multdem1`, `multdem2`, `multdem3`
order of a modulo m	`orderdem` [`a m` [`c`]]
powering algorithm	`pwrdem1a` [`a k m`], `pwrdem1b` [`a k m`], `pwrdem2` [`a k m`]
RSA encryption	`rsa`
square root modulo p	`sqrtdem` [`a p`]
strong pseudoprime test of m base a	`spspdem` [[`a`] `m`]

TABLES

arithmetic functions $\omega(n), \Omega(n), \mu(n), d(n), \phi(n), \sigma(n)$	`arfcntab`
binary quadratic forms	
reduced forms	`qformtab`
forms equivalent to $f(x,y)$	`reduce`
binomial coefficients modulo m	`pascalst`
class numbers	`clanotab`
congruential arithmetic	`cngartab`
discrete logarithms	`indtab`
factorials modulo m	`fctrltab`
Farey fractions	`fareytab`
greatest common divisors	`gcdtab`
indices	`indtab`
intersection of arithmetic progressions	`intaptab`
Jacobi symbols	`jacobtab`
least prime factor	`factab`
linear combinations	`lncomtab`
Lucas functions	`lucastab`
Pascal's triangle modulo m	`pascalst`
powers of a modulo m	`powertab`
representations as sums of powers	`wrg1tab, wrg2tab, wrgcntab`
roots of	
$f(x) \equiv 0 \pmod{p^j}$	`hensel`
$P(x) \equiv 0 \pmod{m}$	`polysolv`

Programs by Section

Warning

Acquisition

This manual and the accompanying programs are available over the internet by using the file transfer program ftp. For detailed instructions see the *Further Resources* at the end of this manual, and particularly the entry relating to CLINT. If you do not have access to ftp, then this material will be sent, free of charge, upon request. Direct your inquiry to

Mathematics Editor
John Wiley & Sons
605 Third Avenue
New York, NY 10158-0012