SOME COMPUTER PROGRAMS REGARDING POLYNOMIALS AND MATRICES

By

BIPUL SYAM PURKAYASTHA

DEPARTMENT OF MATHEMATICS

SUBMITTED

IN

PARTIAL FULFILMENT OF THE REQUIREMENT OF THE DEGREE OF

MASTER OF PHILOSOPHY

To

NORTH-EASTERN HILL UNIVERSITY

SHILLONG

AUGUST, 1989

Dr. VIJAI KUMAR
READER

DEPARTMENT OF MATHEMATICS
North-Eastern Hill University
Bijni Complex, Laitumkhrah
SHILLONG 793 003

*I certify that the dissertation entitled "**Some Programs regarding Polynomials and Matrices**" submitted by Mr. Bipul Syam Purkayastha in partial fulfilment of the requirements for the degree of Master of Philosophy is the outcome of a study undertaken by the candidate. I certify that sources from which ideas have been borrowed are duly referred to.*

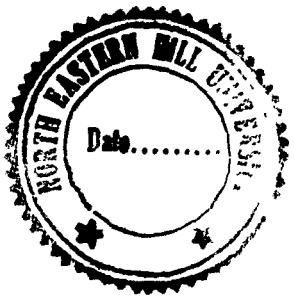
The material in this dissertation has not been presented for the award of a degree in any University before.

This dissertation may be placed before the examiners for evaluation and necessary formalities.

विजय कुमार

SHILLONG
The 31st August 1989

(Vijai Kumar)
Supervisor



Malt

DS
519.77
PUR

ACKNOWLEDGEMENT

The study for this dissertation was undertaken under the guidance of Dr. **Vijai Kumar**. I wish to express my deep gratitude to him for suggesting the problems and guiding me into my first venture into research with a rare sense of patience and understanding. Sir also undertook to type the whole manuscript working late hours, meticulous and caring as he is.

I am thankful to the Department of Mathematics, NEHU and in particular to the Head of the Department Dr. **S.S. Khare** for his constant help and encouragement. I owe special thanks to Prof. **S.N. Dube** for his initial help and to my other teachers for their kind co-operation, particularly those at Mathematics Department.

My colleagues at Shillong College extended their helping hands for the smooth completion of this work. With their constant encouragement my problems and frustrations were more bearable. Special mention must be made of Mr. **Rawat**, the Principal, Shillong College, who also extended his kind co-operation and help.

I would also like to convey my warmest thanks to all the M.Phil & M.Sc. students at NEHU for their cooperation.

Lastly, I'd like to thank all my friends and in particular to Dr. **B.M. Jyrwa** of the Physics Department who gave me access to use the computer there.

Bipul Syam Purkayastha

BIPUL SYAM PURKAYASTHA

CONTENTS

ABSTRACT

In every computer, we have a certain upper limit beyond which an integer cannot be stored in the memory. If we work with real numbers instead of integers, the results are not exact due to round-off errors. If we are dealing with large numbers having, say more than 15 digits, and we want the exact answers, it becomes quite a problem with the computers. Let us assume that the maximum limit for an integer variable in a particular computer is 99999 and certain answer happens to be 837592103563789. Then we can divide the number into groups of five digits each, starting from the right, and store each group of five digits in an element of an integer array, say M. In the above case we shall get M(1)=63789, M(2)=21035, M(3)=87359. Then we can write the following statements in our program:-

```
      I = 3
      PRINT 80, (M (I+1-k), K=1,I)
80    FORMAT (1X, 26I5)
```

The computer will print 87359 21035 63789 in the 1st 15 columns of the line which is correct. But if we proceed in the above manner, there is a difficulty. If the exact

answer happens to be ~~16434 0005 100008~~ 1643400051000 08 the computer will print out

~~164 34 51 8~~ 16434 51 8

in the 1st 15 columns of the printed line. The embedded zeros will be replaced by blanks, thus rendering the answer absurd. Assuming that the correct answer will not have more than 80 digits, and dividing each number into groups of 4 digits each, starting from the right I have developed 5 programs out of which four are the for four fundamental operations — addition, subtraction, multiplication and division for large numbers. The fifth program deals with H.C.F. of large numbers. I have taken adequate precautions in my programs so that the answers and the numbers are printed correctly. When we talk about polynomials in one variable, it is natural to think about the zeros of a polynomial and the coefficients of the polynomials. I have developed some programs for reconstructing a polynomial from its integer zeros. I have extended this problem to the case when the zeros of the required polynomial are rational numbers, but not necessarily integers. These rational numbers may not be given in their lowest terms. I have arranged matters so that the H.C.F. of all the coefficients of the polynomial will be unity and the leading coefficient will be positive.

Next I have considered the inverse problem, that is the coefficients are given as integers and we are to calculate the zeros. My program for this problem will generate not all zeros, but only those zeros which are rational. In case, none of the zeros is rational, the program will print out an appropriate message to this effect.

In the case of a monic polynomial we know that all the rational zeros must be necessarily integers. However, if the polynomial is not monic, we can reduce the polynomial to another in which the leading coefficient will be positive and the H.C.F. of all the coefficients will be unity. If these conditions are satisfied, and if the coefficients of the polynomial happen to be integers, the rational zeros, if any, will be in the form p/q, where (p,q)=1, p is a factor of the last coefficient of the polynomial and q is a factor of the first coefficient. It will be cumbersome to try all the combinations of p and q in the search for rational zeros. A better method, still, is to multiply all zeros of the polynomial by the smallest positive integer K so that the polynomial becomes monic. The polynomial becomes then

$$x^n + a_1 x^{n-1} + a_2 x^{n-2} + - - - - - - - - + a_n$$

where all the a'S are integers. It is easy now to find all rational (integer) zeros, and having found them,

we divide by K and reduce each zero to the lowest terms. Those ideas have led me to write a program which will calculate the smallest possible value of K.

While working with the computer, we face some difficulties with DO loops. If we have a set of variables I, J, K, L, M say 5 in number, where I varies from 1 to 8, J varies from 1 to 7, ..., M varies from 1 to 10 and if we have to evaluate a quantity which depends on all these variables and if we have a restriction that none of the variables can be equal to one another, how shall we write a program? I have written a program to solve this difficulty and also another program, for generating N DO loops and start execution from any set of values. As a byproduct of these two programs I have written another two programs for generating all possible permutations and combinations of the first N natural numbers taking R at a time.

Again, if a square matrix is given, we may be interested in calculating all its characteristic roots. For this purpose we must first obtain its characteristic polynomial .

Assuming the entries of the matrix are integers, I have developed two programs, one for generating the characteristic polynomial of the given square matrix and the other for calculating the characteristic roots of the matrix.

CHAPTER 1

HANDLING OF BIG INTEGERS IN THE COMPUTER

1. INTRODUCTION

WE KNOW THAT COMPUTERS DEAL WITH NUMBERS IN 2 MODES, NAMELY THE INTEGER MODE AND THE REAL MODE. FOR EVERY COMPUTER THERE IS A CERTAIN LIMIT UPTO WHICH NUMBERS CAN BE STORED IN A SINGLE MEMORY LOCATION IN INTEGER MODE. FOR EXAMPLE IN THE FOLLOWING COMPUTERS THE INTEGER LIMITS ARE AS FOLLOWS:-

BURROUGHS 6700-7700 SERIES:-	$2^{39}-1$
DEC 10-20 SERIES:-	$2^{35}-1$
IBM 360-370 SERIES:-	$2^{31}-1$
ICL 1900 SERIES:-	$2^{23}-1$
TDC-316:-	$2^{15}-1$ OR 32767
IBM 1620 SERIES:-	$10^{5}-1$ OR 99999

IN EVERY COMPUTER THERE IS AN UPPER LIMIT AND ALSO A LOWER LIMIT FOR REAL NUMBERS AND THE UPPER LIMIT FOR REAL NUMBERS IS MUCH HIGHER THAN THE LIMIT FOR INTEGERS, FOR EXAMPLE, SOME COMPUTERS CAN DEAL WITH REAL NUMBERS FROM $\pm$0.1E-99 TO $\pm$0.99999999E+99 AND OTHER COMPUTERS CAN DEAL WITH REAL NUMBERS FROM $\pm 10^{-79}$ APPROXIMATELY TO $\pm 10^{75}$ APPROXIMATELY. BUT IT IS WELL KNOWN THAT WHEN WE OPERATE THE COMPUTER IN REAL ARITHMETIC, ROUND-OFF ERRORS ALWAYS ARISE DUE TO THE INTERNAL CONVERSION OF DECIMAL NUMBERS TO THE BINARY SYSTEM

AND AGAIN CONVERSION BACK INTO THE DECIMAL SYSTEM. SO THE ANSWERS OBTAINED BY USING REAL ARITHMETIC ARE NEVER EXACT. IF WE USE DOUBLE PRECISION ARITHMETIC, NO DOUBT, THE ACCURACY INCREASES BUT STILL WE DO NOT OBTAIN EXACT ANSWERS. IT IS NOT THAT COMPUTER SCIENTISTS AND PROGRAMMERS ARE UNAWARE OF THESE LIMITATIONS OF REAL ARITHMETIC. IN FACT, THEY KNOW VERY WELL THAT SUCH DIFFICULTIES ARISE. PROF. N.N.BISWAS IN HIS BOOK ON COMPUTER PROGRAMMING HAS ASKED THE FOLLOWING 2 QUESTIONS:-

1. THE EXACT VALUE OF 18! IS 6402373O5728000. IS IT POSSIBLE TO WRITE A PROGRAM ON IBM 360-370 SERIES TO COMPUTE THIS EXACT VALUE? IF SO, WRITE THE PROGRAM. IF NOT, GIVE REASONS.
2. THE TOTAL NUMBER OF VARIABLES WHICH CAN BE COINED IN FORTRAN IS 551329156. IS IT POSSIBLE TO CALCULATE THIS NUMBER EXACTLY BY WRITING A PROGRAM WORKABLE ON IBM 360-370? IF SO, WRITE THE PROGRAM. IF NOT, TELL WHY THIS PROGRAM CANNOT BE WRITTEN.

FROM THE ABOVE QUESTIONS IT SEEMS THAT THE AUTHOR THINKS THAT THIS PROGRAM CANNOT BE WRITTEN. BUT WE MUST REMEMBER THAT A COMPUTER IS A MOST VERSATILE PIECE OF EQUIPMENT. IT CAN DO ANY CALCULATION WHICH A HUMAN BEING CAN DO. THEREFORE IN THIS CHAPTER I HAVE WRITTEN PROGRAMS FOR THE 4 BASIC ARITHMETICAL OPERATIONS ON NON-NEGATIVE INTEGERS. THESE INTEGERS CAN BE AS BIG AS WE PLEASE.

IN THESE PROGRAMS I HAVE CONSIDERED MAXIMUM 80 DIGITS FOR EVERY INTEGER, BUT THIS LIMIT IS NOT AT ALL FIXED. IF WE INCREASE THE NUMBERS GIVEN IN THE VARIOUS DIMENSION STATEMENTS IN THESE PROGRAMS, THESE PROGRAMS WILL TAKE CARE OF BIGGER INTEGERS ALSO. WE HAVE PROVED THAT NOTHING IS IMPOSSIBLE ON A COMPUTER AND WE CAN OBTAIN 100% ACCURATE RESULTS, CORRECT TO THE LAST UNIT, BY USING INTEGER AND ONLY INTEGER ARITHMETIC. THIS IS A FEAT WHICH IS NOT POSSIBLE BY USING REAL ARITHMETIC. WE MUST REMEMBER THAT WHEN WE LEARNT THE ADDITION AND MULTIPLICATION TABLES IN OUR EARLIEST SCHOOLING DAYS WHILE LEARNING ARITHMETIC, WE LEARNT BY HEART ONLY THESE TABLES UPTO 10 BY 10. NOTWITHSTANDING THIS FACT, WITH THE HELP OF THESE TABLES ONLY, WE CAN, AT LEAST THEORETICALLY, PERFORM THE 4 BASIC ARITHMETICAL OPERATIONS ON ANY INTEGERS, HOWEVER LARGE, IN THE HUMAN DOMAIN. IF THIS IS POSSIBLE FOR A HUMAN BEING, WHY NOT FOR A COMPUTER, WHICH HAS SUCH A VAST MEMORY AND TREMENDOUS SPEED OF CALCULATION?

2. APPLICATIONS OF THESE PROGRAMS

IN NUMBER THEORY, WE COME ACROSS CALCULATIONS OF SUCH TYPE WHERE WE CANNOT TOLERATE ANY ERROR EVEN IN THE LAST UNIT OF THE RESULT. WHEN WE DEAL WITH STATISTICS REGARDING THE POPULATIONS OF COUNTRIES OR REGARDING MONEY MATTERS, WE COME ACROSS VERY BIG NUMBERS, WHICH WILL NORMALLY OVERFLOW THE MEMORY OF ANY COMPUTER.

EVEN SO, WE REQUIRE OUR RESULTS TO BE EXTREMELY ACCURATE. FOR EXAMPLE, IF WE WISH TO PREPARE A CHART SHOWING THE POPULATIONS OF VARIOUS COUNTRIES, WE MAY LIKE TO ADD THOSE FIGURES BY A COMPUTER TO CALCULATE THE TOTAL POPULATION OF THE WORLD. SIMILARLY IN BANKS, INSURANCE COMPANIES, ETC., WE CONSTANTLY DEAL WITH RUPEES AND PAISE. EVEN IF WE MAKE A MISTAKE OF A FEW PAISE IN CALCULATING THE BALANCES OR THE FIGURES OF INTEREST, OUR ACCOUNTS WILL GO WRONG. NOW THAT COMPUTERS ARE BEING USED IN THESE ORGANISATIONS ON A LARGE SCALE, THESE PROGRAMS CAN FIND APPLICATIONS THERE. WHEN WE DEAL WITH EVALUATION OF DETERMINANTS, MATRIX OPERATIONS, SOLUTION OF LINEAR SIMULTANEOUS EQUATIONS, CALCULATION OF THE CHARACTERISTIC ROOTS OF SQUARE MATRICES, FINDING THE ZEROS OF POLYNOMIALS, ETC., IF WE CALCULATE IN REAL MODE, THE ANSWERS WILL NEVER BE EXACT DUE TO ROUND-OFF ERRORS. IN THIS DISSERTATION ITSELF I SHALL TAKE SUCH PROBLEMS IN THE SUBSEQUENT CHAPTERS. TO OBTAIN CORRECT RESULTS WE HAVE TO USE ONLY INTEGER ARITHMETIC AND WHEN WE COME TO VULGAR FRACTIONS, WE HAVE TO STORE THEIR NUMERATORS AND DENOMINATORS IN SEPARATE MEMORY LOCATIONS. EVEN WITH SMALL MATRICES, THESE NUMBERS BECOME BIGGER AND BIGGER VERY FAST AS THE CALCULATIONS PROCEEDS. THEY SOON OVERFLOW THE INTEGER LIMIT OF ANY COMPUTER. SO IN ORDER TO OBTAIN SUCCESS IN SUCH CALCULATIONS WE HAVE TO USE THESE PROGRAMS WHICH FOLLOW. SUPPOSE THAT WE ARE INTERESTED IN CALCULATING THE RANK OF A MATRIX. IF WE ENTRUST OUR CALCULATIONS TO REAL

ARITHMETIC, IT IS QUITE POSSIBLE THAT SOME SUB-DETERMINANTS OF THE MATRIX, WHICH ARE ACTUALLY ZERO, WILL CALCULATE TO SOME VERY SMALL BUT NON-ZERO NUMBERS DUE TO THE ROUND-OFF ERRORS INHERENT IN REAL ARITHMETIC. THEREFORE THE RANK OF THE MATRIX WILL BE WRONGLY CALCULATED AND ALL OUR WORK WILL BECOME USELESS. ON THE OTHER HAND, IF WE CONFINE OUR CALCULATIONS TO INTEGER ARITHMETIC, WHILE DIVIDING, WE SHALL HAVE TO STORE OUR NUMERATORS AND DENOMINATORS IN DIFFERENT MEMORY LOCATIONS AND EVEN AFTER REDUCING ALL FRACTIONS TO LOWEST TERMS AFTER EACH AND EVERY CALCULATION, THE NUMBERS INVOLVED WILL GROW AT AN EXPONENTIAL RATE AND SOON THEY WILL CROSS THE LIMITS SET FOR AN INTEGER VARIABLE IN ANY COMPUTER. THEREFORE THE ONLY POSSIBLE ALTERNATIVE IS TO USE THESE PROGRAMS WHICH I HAVE DEVELOPED IN THIS CHAPTER. IN FACT, I HAVE NOT PHYSICALLY JOINED THESE PROGRAMS TO THOSE PROGRAMS WHICH I HAVE DEVELOPED IN THE OTHER CHAPTERS, BUT WHENEVER OCCASION COMES, WE CAN DO IT IMMEDIATELY AND THUS GET PROGRAMS WHICH WILL WORK SATISFACTORILY IN ALL POSSIBLE SITUATIONS.

3. ADDITION OF LARGE NUMBERS

WHEN WE HAVE TO ADD 2 BIG NUMBERS, WE DIVIDE EACH NUMBER INTO PIECES CONTAINING A FIXED NUMBER OF DIGITS STARTING FROM THE RIGHT. THE NUMBER OF DIGITS IN EACH PIECE WILL DEPEND ON THE PARTICULAR COMPUTER ON WHICH WE ARE WORKING. FOR EXAMPLE, IN IBM 1620 SERIES COMPUTER, THE UPPER LIMIT IS 99999. SO WE SHALL TAKE PIECES OF 5

DIGITS EACH. IN OUR COMPUTER (HCL WORKHORSE-II) THE UPPER LIMIT FOR AN INTEGER VARIABLE IS 32767; SO WE USE PIECES OF 4 DIGITS EACH. WE STORE THESE PIECES SEPARATELY IN 1-DIMENSIONAL ARRAYS CALLED L AND M. WE DECLARE L AND M TO BE ARRAYS OF SIZE 20 EACH; THUS WE CAN TAKE CARE OF MAXIMUM 80 DIGITS IN EACH INTEGER. THEN WE ADD M(1) TO L(1), M(2) TO L(2), M(3) TO L(3), ETC. WE TAKE CARE OF POSSIBLE CARRYING DIGITS AT EACH STEP. WE TAKE PARTICULAR PRECAUTION THAT NO NUMBER STORED IN A SINGLE MEMORY LOCATION BECOMES 10000 OR MORE. (ALTHOUGH THE ACTUAL UPPER LIMIT IS HIGHER, NAMELY 32767, STILL WE DO NOT ALLOW ANY NUMBER TO BECOME EVEN 10000.) IF THE PIECES OF THE SUM ARE STORED IN N(1), N(2), N(3), ETC., WE PRINT THEM BACKWARDS IN A SINGLE LINE. WHILE PRINTING IF SOME PIECE CONTAINS LESS THAN 4 DIGITS, IN NORMAL PRINTING THE MACHINE WILL LEAVE A CERTAIN NUMBER OF BLANKS. TO AVOID THESE BLANKS WE USE A SPECIAL SUBROUTINE SUBPROGRAM WHICH FILLS UP THESE BLANKS BY ZEROS USING H-FORMAT. WE DO NOT DO THIS IN THE LEFTMOST PIECE, SO THAT ALL NUMBERS ARE PRINTED IN A TRADITIONAL WAY. THE PROGRAM IS GIVEN ON THE NEXT PAGE, TOGETHER WITH ALL THE SUBROUTINES USED. IN SUITABLE LINES WE HAVE INSERTED COMMENT STATEMENTS ALSO, SO THAT THE LOGIC OF THE PROGRAM CAN BECOME CLEARER. IN THE BEGINNING OF THE PROGRAM WE HAVE WRITTEN THE FOLLOWING DIMENSION STATEMENT:-

```
DIMENSION L(20),M(20),N(20)
```

IT IS POSSIBLE THAT INDIVIDUALLY THE TOTAL NUMBER OF PIECES IN THE

2 NUMBERS MAY BE WITHIN 20, BUT ON ADDITION THE NUMBER OF PIECES IN THE SUM MAY BECOME 21. IN SUCH A CASE THE OPERATING SYSTEM OF THE COMPUTER WILL CERTAINLY PRINT OUT AN ERROR MESSAGE WHEN WE SHALL ASK SOME NUMBER TO BE STORED IN N(21). TO AVOID SUCH A CIRCUMSTANCE, IN THE PROGRAM ITSELF WE HAVE WRITTEN SUCH STATEMENTS THAT WHEN THE NUMBER OF PIECES IN THE SUM IS LIKELY TO EXCEED 20, THE PROGRAM DOES NOT PROCEED FURTHER, BUT PRINTS THE FOLLOWING MESSAGE USING H-FORMAT:-

THE SUM OF THE 2 NUMBERS HAS MORE THAN 80 DIGITS

BY THIS TECHNIQUE WE AVOID THE ERROR MESSAGE WHICH THE OPERATING SYSTEM WOULD OTHERWISE PRINT.

//

```
      DIMENSION L(20),M(20),N(20)
6     WRITE(1,34)
34    FORMAT(' TYPE THE NUMBERS OF PIECES IN THE 2 INTEGERS TO BE',
     11X,'ADDED IN FORMAT ( 2I2 )' )
62    READ ( 1 , 61 ) I , J
61    FORMAT( 2I2 )
      IF ( I .EQ. 0 .AND. J .EQ. 0 ) STOP
      IF ( I .GT. 0 .AND. J .GT. 0 ) GO TO 181
      WRITE(1,89)
89    FORMAT( 1X, 'DATA ILLEGAL, PLEASE TYPE AGAIN' )
      GO TO 62
181   IF ( I .LE. 20 .AND. J .LE. 20 ) GO TO 205
      IF ( I .LE. 20 .OR. J .LE. 20 ) GO TO 207
      WRITE ( 1, 90 )
90    FORMAT(1X,'BOTH NUMBERS TOO LARGE, PLEASE TYPE AGAIN' )
      GO TO 62
207   IF ( I .LE. 20 ) GO TO 305
      WRITE ( 1 , 91 )
91    FORMAT(1X, 'FIRST NUMBER TOO LARGE, PLEASE TYPE AGAIN' )
      GO TO 62
305   WRITE ( 1 , 92 )
92    FORMAT ( 1X , 'SECOND NUMBER TOO LARGE, PLEASE TYPE AGAIN' )
      GO TO 62
205   WRITE ( 1 , 109 )
109   FORMAT ( 1X , 'TYPE THE FIRST NUMBER IN FORMAT ( 20I4 )' )
      READ ( 1 , 86 ) ( L ( I + 1 - K ) , K = 1 , I )
      WRITE ( 1 , 110 )
110   FORMAT ( 1X , 'TYPE THE SECOND NUMBER IN FORMAT ( 20I4 )' )
      READ ( 1 , 86 ) ( M ( J + 1 - LL ) , LL = 1 , J )
86    FORMAT ( 20I4 )
      IND = 0
      CALL ADDN ( L , M , N , I , J , KK , IND )
      IF ( IND .NE. 1 ) GO TO 1000
      WRITE ( 1 ,1004 )
1004  FORMAT ( 1X , 'THE SUM HAS MORE THAN 80 DIGITS' )
      GO TO 6
1000  WRITE ( 1, 654 ) L ( I )
654   FORMAT ( 1X , I4$ )
      IF ( I .EQ. 1 ) GO TO 245
      DO 267 II = 2 , I
267   CALL PRINT ( L ( I + 1 - II ) )
245   WRITE ( 1 ,700 )
700   FORMAT ( 40X )
      WRITE ( 1 , 654 ) M ( J )
      IF ( J .EQ. 1 ) GO TO 258
      DO 268 JJ = 2 , J
268   CALL PRINT ( M ( J + 1 - JJ ) )
258   WRITE ( 1 , 700 )
      WRITE ( 1, 654 ) N ( KK )
```

```
      IF ( KK .EQ. 1 ) GO TO 279
      DO 389 KKK = 2 , KK
389   CALL PRINT ( N ( KK + 1 - KKK ) )
279   WRITE ( 1 , 700 )
      GO TO 6
      END
      SUBROUTINE ADDN(L,M,N,LL,MM, NN , IND )
      INTEGER CARRY
      DIMENSION L(20),M(20),N(20)
C     WHEN WE START THE ADDITION, THERE IS NOTHING TO CARRY,  SO WE SET
C     CARRY AS ZERO
      CARRY = 0
C     COMPARE THE NUMBERS OF PIECES IN THE 2 NUMBERS
      IF(LL-MM)300,276,202
C     THE FIRST NUMBER HAS MORE PIECES THAN THE SECOND
202   LIMIT=MM
      IN=MM+1
      DO 244 II=IN,LL
244   N(II)=L(II)
      NN=LL
      GO TO 400
276   LIMIT=MM
      NN=LIMIT
      GO TO 400
300   LIMIT=LL
      IN=LL+1
      DO 344 II=IN,MM
344   N(II)=M(II)
      NN=MM
C     ADD THE NUMBERS PIECE BY PIECE, TAKING CARE THAT THE UPPER LIMIT
C     OF 9999 IS NOT CROSSED
400   DO 522 I=1,LIMIT
      KTEST=9999-M(I)-CARRY
      IF(L(I).NE.9999.OR.KTEST.NE.-1)GO TO 464
      N(I)=9999
432   CARRY=1
      GO TO 522
464   IF(L(I).LE.KTEST)GO TO 498
      N(I)=L(I)-KTEST-1
      GO TO 432
498   N(I)=9999-(KTEST-L(I))
      CARRY=0
522   CONTINUE
C     IF THE LAST PIECE OF THE SUM IS GREATER THAN 9999, THEN THE CARRY
C     IS UNITY, OTHERWISE THE CARRY IS ZERO
      IF(CARRY.NE.0)GO TO 558
      RETURN
C     IF CARRY = 1, EITHER ADD CARRY TO THE NEXT PIECE, OR INCREASE THE
C     NUMBER OF PIECES BY 1
558   IF(NN.NE.LIMIT)GO TO 601
```

```
582     NN=NN+1
        IF ( NN .GT. 20 ) GO TO 800
        N(NN)=1
        RETURN
601     LIMIT=LIMIT+1
        DO 644 I=LIMIT,NN
        IF(N(I).NE.9999)GO TO 676
        N(I)=0
644     CONTINUE
        GO TO 582
676     N(I)=N(I)+1
        RETURN
800     IND = 1
        RETURN
        END
        SUBROUTINE PRINT(M)
        IF(M.LT.10)GO TO 1
        IF(M.LT.100)GO TO 2
        IF(M.LT.1000)GO TO 3
        WRITE(1,70)M
70      FORMAT(1X,I4$)
        RETURN
3       WRITE(1,71)M
71      FORMAT(1X,1H0,I3$)
        RETURN
2       WRITE(1,72)M
72      FORMAT(1X,2H00,I2$)
        RETURN
1       WRITE(1,73)M
73      FORMAT(1X,3H00C,I1$)
        RETURN
        END

//
```

WE OBSERVE THAT IN STATEMENT NO. 654 OF THE MAIN PROGRAM (FORMAT STATEMENT) WE HAVE WRITTEN A CURRENCY SYMBOL (₹) AFTER I4. IN STATEMENTS NOS. 70, 71, 72, 73 OF THE SUBROUTINE SUBPROGRAM CALLED PRINT ALSO (ALL FORMAT STATEMENTS) WE HAVE PLACED THE ABOVE SYMBOL AFTER THE RELEVANT I-FIELD. THE REASON FOR WRITING THIS SYMBOL IS THAT IN OUR COMPUTER THIS IS THE PROPER SYMBOL TO ENSURE THAT THE PRINTER DOES NOT GO TO A NEW LINE AFTER PRINTING THE VALUE OF THE LAST VARIABLE. IF WE WOULD NOT HAVE WRITTEN THE SYMBOL $ IN THE FORMAT STATEMENTS, ALL THE PIECES OF THE VARIOUS NUMBERS TO BE PRINTED IN ONE LINE WOULD HAVE BEEN PRINTED IN DIFFERENT LINES, THUS MAKING THE FINAL PRINT-OUT MEANINGLESS. HOWEVER, $ IS NOT THE NORMAL SYMBOL TO BE USED IN SUCH CASES. IN MOST COMPUTERS THIS OBJECTIVE IS ACHIEVED BY WRITING THE CARRIAGE CONTROL CHARACTER (1H+) OR ('+') IN THE BEGINNING OF A FORMAT STATEMENT. BUT OUR COMPUTER DOES NOT ACCEPT THIS CARRIAGE CONTROL CHARACTER. SO WE HAD TO USE THE $ SYMBOL INSTEAD OF (1H+).

4. SUBTRACTION OF LARGE NUMBERS

SIMILARLY FOR SUBTRACTION OF A LARGE NUMBER FROM ANOTHER LARGE NUMBER WE DIVIDE THE MINUEND AND THE SUBTRAHEND INTO PIECES JUST LIKE IN THE CASE OF ADDITION. BEFORE WE START THE SUBTRACTION, WE MUST KNOW WHICH OF THE 2 NUMBERS, IF ANY, IS BIGGER. FOR THIS PURPOSE FIRST OF ALL WE COMPARE THE NUMBER OF PIECES IN THE MINUEND WITH THE NUMBER OF PIECES IN THE SUBTRAHEND. IF THE FORMER NUMBER IS BIGGER, THEN WE DECIDE TO SUBTRACT THE SUBTRAHEND FROM THE MINUEND. IF THE LATTER NUMBER IS BIGGER, WE DECIDE TO SUBTRACT THE

//

MINUEND FROM THE SUBTRAHEND. IF BOTH NUMBERS ARE EQUAL, WE COMPARE THE PIECES OF THE MINUEND AND THE SUBTRAHEND STARTING FROM THE LEFTMOST PIECES. THIS COMPARISON CAN BE DONE VERY EASILY USING A DO LOOP. IF AT ANY STAGE WE FIND THAT SOME PIECE OF THE MINUEND IS BIGGER, WE DECIDE TO SUBTRACT THE SUBTRAHEND FROM THE MINUEND, AND IF WE FIND OTHERWISE, WE DECIDE THE REVERSE. HOWEVER, IF WE FIND THAT PIECE BY PIECE ALL THE PIECES OF THE MINUEND ARE EQUAL TO THE CORRESPONDING PIECES OF THE SUBTRAHEND, WE CONCLUDE THAT BOTH THE NUMBERS ARE EQUAL. IN THIS CASE WE SET AN INDEX EQUAL TO ZERO AND DEFINE THE REMAINDER AS ZERO. FOR THIS PURPOSE WE SAY THAT THE REMAINDER HAS JUST ONE PIECE AND THAT PIECE IS ZERO. IF OUR DECISION IS TO SUBTRACT THE SUBTRAHEND FROM THE MINUEND, WE SET THE INDEX AS ZERO AND START THE CALCULATIONS. IF OUR DECISION IS THE REVERSE, WE SET THE INDEX AS UNITY AND START THE SUBTRACTION IN THE REVERSE ORDER. THE REASON FOR THIS APPROACH IS THAT, FINALLY, IF INDEX HAPPENS TO BE UNITY, WE PRINT A NEGATIVE SIGN IN THE BEGINNING OF THE REMAINDER, WHICH IS THE ANSWER. THIS HAPPENS IF THE SUBTRAHEND IS GREATER THAN THE MINUEND. IF THE SUBTRAHEND IS LESS THAN OR EQUAL TO THE MINUEND, THE PROGRAM SETS THE INDEX TO ZERO AND IN THIS CASE WE DO NOT PRINT ANY NEGATIVE SIGN IN THE BEGINNING OF THE REMAINDER. SO IN ALL CASES THE ANSWER PRINTED IS ACCORDING TO THE TRADITIONAL FASHION. DURING THE SUBTRACTION, WE CONSTANTLY TAKE CARE OF THE BORROWING FIGURE, IF ANY. WE TAKE PARTICULAR PRECAUTIONS THAT NO NUMBER STORED IN A SINGLE MEMORY LOCATION BECOMES NEGATIVE OR GREATER THAN 9999. WE PRINT THE

ANSWER JUST LIKE THE CASE OF ADDITION. WE ALSO TAKE CARE THAT IF SOME OF THE LEFTMOST PIECES IN THE REMAINDER HAPPEN TO BE ZERO, THEN THESE PIECES ARE NOT PRINTED AT ALL. FOR EXAMPLE, IF N(5) = N(6) = N(7) = 0 BUT N(4) IS NOT ZERO, WE SHALL PRINT ONLY FROM N(4) TO N(1) IN ONE LINE. THE PROGRAM OF SUBTRACTION, TOGETHER WITH ALL THE SUBROUTINES USED, IS GIVEN ON THE SUCCEEDING PAGES.

//

```
      DIMENSION L ( 20 ) , M ( 20 ) , N ( 20 )
6     WRITE ( 1 , 34 )
34    FORMAT ( 1X , 'TYPE THE NUMBERS OF PIECES IN THE MINUEND AND'
     11X, 'THE SUBTRAHEND IN FORMAT ( 2I2 )' )
62    READ ( 1 , 61 ) I , J
61    FORMAT ( 2I2 )
      IF ( I .EQ. 0 .AND. J .EQ. 0 ) STOP
      IF ( I .GT. 0 .AND. J .GT. 0 ) GO TO 181
      WRITE ( 1 , 89 )
89    FORMAT ( 1X , 'DATA ILLEGAL, PLEASE TYPE AGAIN' )
      GO TO 62
181   IF ( I .LE. 20 .AND. J .LE. 20 ) GO TO 205
      IF ( I .LE. 20 .OR. J .LE. 20 ) GO TO 207
      WRITE ( 1 , 90 )
90    FORMAT ( 1X , 'BOTH NUMBERS TOO LARGE, PLEASE TYPE AGAIN' )
      GO TO 62
207   IF ( I .LE. 20 ) GO TO 305
      WRITE ( 1 , 91 )
91    FORMAT ( 1X , 'MINUEND TOO LARGE, PLEASE TYPE AGAIN' )
      GO TO 62
305   WRITE ( 1 , 92 )
92    FORMAT ( 1X , 'SUBTRAHEND TOO LARGE, PLEASE TYPE AGAIN' )
      GO TO 62
205   WRITE ( 1 , 109 )
109   FORMAT ( 1X , 'TYPE THE MINUEND IN FORMAT ( 20I4 )' )
      READ ( 1 , 86 ) ( L ( I + 1 - I ) , I = 1 , I )
      WRITE ( 1 , 110 )
110   FORMAT ( 1X , 'TYPE THE SUBTRAHEND IN FORMAT ( 20I4 )' )
      READ ( 1 , 86 ) ( M ( J + 1 - KK ) , KK = 1 , J )
86    FORMAT ( 20I4 )
      CALL BIGSUB ( L , M , N , I , J , NN , INDEX )
      WRITE ( 1 , 654 ) L ( I )
654   FORMAT ( 1X , I4$ )
      IF ( I .EQ. 1 ) GO TO 245
      DO 267 I2 = 2 , I
267   CALL PRINT ( L ( I + 1 - I2 ) )
245   WRITE ( 1 , 700 )
700   FORMAT ( 40X )
      WRITE ( 1 , 654 ) M ( J )
      IF ( J .EQ. 1 ) GO TO 258
      DO 268 JJ = 2 , J
268   CALL PRINT ( M ( J + 1 - JJ ) )
258   WRITE ( 1 , 700 )
      IF ( INDEX .EQ. 1 ) GO TO 234
      WRITE ( 1 , 654 ) N ( NN )
      IF ( NN .EQ. 1 ) GO TO 279
      DO 389 KKK = 2 , NN
389   CALL PRINT ( N ( NN + 1 - KKK ) )
```

```
279   WRITE ( 1 , 700 )
      GO TO 6
234   IF ( N ( NN ) .LT. 10 ) GO TO 99
      IF ( N ( NN ) .LT. 100 ) GO TO 102
      IF ( N ( NN ) .LT. 1000 ) GO TO 103
      WRITE ( 1 , 800 ) N ( NN )
      GO TO 500
99    WRITE ( 1 , 801 ) N ( NN )
      GO TO 500
102   WRITE ( 1 , 802 ) N ( NN )
      GO TO 500
103   WRITE ( 1 , 803 ) N ( NN )
803   FORMAT ( 1X , 1H-, I3$ )
802   FORMAT ( 1X , 1H-, I2$ )
801   FORMAT ( 1X , 1H-, I1$ )
800   FORMAT ( 1X , 1H-, I4$ )
500   IF ( NN .EQ. 1 ) GO TO 765
      DO 766 NNN = 2 , NN
766   CALL PRINT ( N ( NN + 1 - NNN ) )
765   WRITE ( 1 , 700 )
      GO TO 6
      END
      SUBROUTINE BIGSUB(L,M,N,LL,MM,NN,INDEX)
      INTEGER CARRY
      DIMENSION L(20),N1(20),M(20),N(20)
      CARRY=0
C     COMPARE THE NUMBERS OF PIECES IN THE 2 NUMBERS
      IF(LL-MM)590,681,202
C     FOR SUBTRAHEND > MINUEND, PUT INDEX = 1, OTHERWISE INDEX = 0
202   INDEX = 0
      DO 244 I=1,MM
244   N1(I)=M(I)
      DO 288 I=1,LL
288   N(I)=L(I)
      NN=LL
320   KK=MM
C     SUBTRACT EACH PIECE OF 1 NUMBER FROM THE CORRESPONDING PIECE OF
C     THE OTHER NUMBER
350   DO 488 I=1,KK
      IDIFF=N(I)-N1(I)
      IF(IDIFF)451,421,386
386   N(I)=IDIFF-CARRY
      CARRY=0
      GO TO 488
421   IF(CARRY.NE.0)GO TO 451
      N(I)=0
      GO TO 488
451   N(I)=IDIFF+1-CARRY+9999
      CARRY=1
488   CONTINUE
```

```
      IF(KK.EQ.NN)GO TO 754
      IF(CARRY.NE.0)GO TO 512
      RETURN
512   IF(KK+1.EQ.NN)GO TO 553
      I1=NN-1
      II=KK+1
      DO 533 I=II,I1
      IF(N(I).EQ.0)GO TO 533
      N(I)=N(I)-1
      RETURN
533   N(I)=9999
553   N(NN)=N(NN)-1
      GO TO 754
590   INDEX=1
      DO 622 I=1,MM
622   N(I)=M(I)
      DO 655 I=1,LL
655   N1(I)=L(I)
      KK=LL
      NN=MM
      GO TO 350
C     THE NUMBERS OF PIECES BEING EQUAL, COMPARE CORRESPONDING PIECES
C     STARTING FROM THE LEFT
681   DO 711 I=1,MM
      IF(L(MM+1-I)-M(MM+1-I))590,711,202
711   CONTINUE
      INDEX = 0
      NN=1
      N(1)=0
      RETURN
754   IF(NN.EQ.1)RETURN
      MN1=NN-1
C     TEST WHETHER LEFTMOST PIECE(S) ARE ZERO
      DO 888 I=1,MN1
      IF(N(NN).NE.0) RETURN
888   NN=NN-1
      RETURN
      END
      SUBROUTINE PRINT(M)
      IF(M.LT.10)GO TO 1
      IF(M.LT.100)GO TO 2
      IF(M.LT.1000)GO TO 3
      WRITE(1,70)M
70    FORMAT(1X,I4$)
      RETURN
3     WRITE(1,71)M
71    FORMAT(1X,1HO,I3$)
      RETURN
2     WRITE(1,72)M
```

```
72      FORMAT(1X,2H00,I2$)
        RETURN
1       WRITE(1,73)M
73      FORMAT(1X,3H000,I1$)
        RETURN
        END

//
```

5. MULTIPLICATION OF LARGE NUMBERS

FOR MULTIPLICATION WE DIVIDE THE MULTIPLICAND INTO PIECES JUST LIKE ADDITION AND DIGITISE THE MULTIPLIER. FOR EXAMPLE, IF THE MULTIPLIER IS 1259067823, WE DEFINE A 1-DIMENSIONAL ARRAY D SO THAT D(1) = 3, D(2) = 2, D(3) = 8, ... , D(10) = 1. WE MULTIPLY THE MULTIPLICAND BY EACH DIGIT SEPARATELY. FOR THIS PURPOSE WE USE ANOTHER SUBROUTINE SUBPROGRAM CALLED MULT. WE CREATE A TABLE OF 8 MULTIPLES OF THE MULTIPLICAND BY 2,3,4,...,9. WE TAKE A PRECAUTION THAT, FOR EXAMPLE, IF THE MULTIPLIER DOES NOT CONTAIN THE DIGIT 4 AT ALL, WE DO NOT STORE THE PRODUCT OF THE MULTIPLICAND BY 4. IF A PARTICULAR DIGIT IN THE MULTIPLIER IS 0, WE DO NOTHING. IF A PARTICULAR DIGIT IN THE MULTIPLIER IS 1, WE TAKE THE MULTIPLICAND AS SUCH. WE USE ANOTHER SUBROUTINE CALLED SHIFT TO SHIFT THE PRODUCT OF THE MULTIPLICAND BY A SINGLE DIGIT OF THE MULTIPLIER TO THE LEFT BY THE REQUIRED NUMBER OF PLACES, AS WE NORMALLY DO IN THE HUMAN DOMAIN. THEN WE ADD THE PARTIAL PRODUCTS USING THE SAME SUBROUTINE CALLED ADDN, WHICH WE HAVE ALREADY USED IN THE ADDITION PROGRAM FOR 2 BIG NUMBERS. THROUGHOUT THIS PROCESS WE KEEP A CONSTANT WATCH ON THE TOTAL NUMBER OF DIGITS OBTAINED DURING MULTIPLICATION BY A SINGLE DIGIT, SHIFTING OR ADDING. IF WE FIND THAT THIS NUMBER IS GOING TO BECOME GREATER THAN 80, WE IMMEDIATELY SET AN INDICATOR IND AS UNITY AND RETURN TO THE MAIN PROGRAM. THE MAIN PROGRAM THEN PRINTS AN APPROPRIATE MESSAGE AND ABORTS THE JOB. HOWEVER, THIS LIMIT 80 IS NOT FIXED AND IT CAN BE INCREASED AS DESIRED. WE PRINT THE ANSWER WITH EMBEDDED ZEROS, IF ANY, INSTEAD

//

OF BLANKS, THROUGH THE USE OF THE SAME SUBROUTINE CALLED PRINT, WHICH WE HAVE BEEN USING IN THE OTHER PROGRAMS. THE MAIN PROGRAM FOR MULTIPLICATION, TOGETHER WITH THE SUBROUTINES USED (INTEG, MULT, SHIFT, ADDN AND PRINT) IS GIVEN ON THE SUCCEEDING PAGES.

//

```
      DIMENSION L ( 20 ) , M ( 20 ) , N ( 20 )
6     WRITE ( 1 , 34 )
34    FORMAT ( 1X , 'TYPE THE NUMBERS OF PIECES IN THE MULTIPLICAND',
     11X , 'AND THE MULTIPLIER IN FORMAT(2I2)' )
62    READ ( 1 , 61 ) I , J
61    FORMAT ( 2I2 )
      IF ( I .EQ. 0 .AND. J .EQ. 0 ) STOP
      IF ( I .GT. 0 .AND. J .GT. 0 ) GO TO 181
      WRITE ( 1 , 89 )
89    FORMAT ( 1X , 'DATA ILLEGAL, PLEASE TYPE AGAIN' )
      GO TO 62
181   IF ( I .LE. 20 .AND. J .LE. 20 ) GO TO 205
      IF ( I .LE. 20 .OR. J .LE. 20 ) GO TO 207
      WRITE ( 1 , 90 )
90    FORMAT ( 1X , 'BOTH NUMBERS TOO LARGE, PLEASE TYPE AGAIN' )
      GO TO 62
207   IF ( I .LE. 20 ) GO TO 305
      WRITE ( 1 , 91 )
91    FORMAT ( 1X , 'MULTIPLICAND TOO LARGE, PLEASE TYPE AGAIN' )
      GO TO 62
305   WRITE ( 1 , 92 )
92    FORMAT ( 1X , 'MULTIPLIER TOO LARGE, PLEASE TYPE AGAIN' )
      GO TO 62
205   WRITE ( 1 , 109 )
109   FORMAT ( 1X , 'TYPE THE MULTIPLICAND IN FORMAT ( 20I4 )' )
      READ ( 1 , 86 ) ( L ( I + 1 - II ) , II = 1 , I )
      WRITE ( 1 , 110 )
110   FORMAT ( 1X , 'TYPE THE MULTIPLIER IN FORMAT ( 20I4 )' )
      READ ( 1 , 86 ) ( M ( J + 1 - JJ ) , JJ = 1 , J )
86    FORMAT ( 20I4 )
      IND = 0
      CALL BIGMUL ( L , I , M , J , N , K , IND )
      IF ( IND .NE. 1 ) GO TO 1000
      WRITE ( 1 , 1004 )
1004  FORMAT ( 1X , 'THE PRODUCT HAS MORE THAN 80 DIGITS' )
      GO TO 6
1000  WRITE ( 1 , 654 ) L ( I )
654   FORMAT ( 1X , I4$ )
      IF ( I .EQ. 1 ) GO TO 245
      DO 267 I2 = 2 , I
267   CALL PRINT ( L ( I + 1 - I2 ) )
245   WRITE ( 1 , 700 )
700   FORMAT ( 40X )
      WRITE ( 1 , 654 ) M ( J )
      IF ( J .EQ. 1 ) GO TO 258
      DO 268 JJ = 2 , J
268   CALL PRINT ( M ( J + 1 - JJ ) )
258   WRITE ( 1 , 700 )
      WRITE ( 1 , 654 ) N ( K )
      IF ( K .EQ. 1 ) GO TO 279
```

```
      DO 389 KKK = 2 , K
389   CALL PRINT ( N ( K + 1 - KKK ) )
279   WRITE ( 1 , 700 )
      GO TO 6
      END
      SUBROUTINE BIGMUL(A,I,B,J,C,M,IND)
      INTEGER A(20),IPROD(20),B(20),C(20),D(80),INP(9),ISP(9,20),C1(20)
      IS= - 1
      C ( 1 ) = 0
      M = 1
      DO 222 K = 2 , 9
222   INP ( K ) = 0
C     DIGITISE THE MULTIPLIER
      CALL INTEG(B,J,D,IDN)
      DO 555 MM=1,IDN
C     IS DENOTES THE NUMBER OF PLACES BY WHICH WE HAVE TO SHIFT THE
C     PRODUCT
      IS = IS + 1
      IF ( D ( MM ) - 1 ) 555 , 386 , 242
242   IF ( INP ( D ( MM ) ) .NE. 0 ) GO TO 320
      CALL MULT(A,D(MM),IPROD,I,N1, IND )
      IF(IND.EQ.1)RETURN
C     MAKE A TABLE OF 8 MULTIPLES OF THE MULTIPLICAND
      DO 288 K = 1 , N1
288   ISP ( D ( MM ) , K ) = IPROD ( K )
      INP ( D ( MM ) ) = N1
      GO TO 401
320   N1 = INP ( D ( MM ) )
      DO 355 K = 1 , N1
355   IPROD ( K ) = ISP ( D ( MM ) , K )
      GO TO 401
386   DO 422 K = 1 , I
422   IPROD ( K ) = A ( K )
      N1 = I
401   IF ( N1 .EQ. 1 .AND. IPROD ( 1 ) .EQ. 0 ) GO TO 555
      CALL SHIFT ( IPROD , N1 , IS , IND )
      IF ( IND .EQ. 1 )RETURN
      CALL ADDN(C,IPROD,C1,M,N1,M1,IND )
      IF ( IND .EQ. 1 ) RETURN
      M = M1
      DO 488 MK = 1 , M
488   C ( MK ) = C1 ( MK )
555   CONTINUE
      RETURN
      END
      SUBROUTINE ADDN(L,M,N,LL,MM, NN , IND )
      INTEGER CARRY
      DIMENSION L(20) M(20),N(20)
      CARRY = 0
      IF(LL-MM)300,276,202
202   LIMIT=MM
```

```
      IN=MM+1
      DO 244 II=IN,LL
244   N(II)=L(II)
      NN=LL
      GO TO 400
276   LIMIT=MM
      NN=LIMIT
      GO TO 400
300   LIMIT=LL
      IN=LL+1
      DO 344 II=IN,MM
344   N(II)=M(II)
      NN=MM
400   DO 522 I=1,LIMIT
      ITEST=9999-M(I)-CARRY
      IF(L(I).NE.9999.OR.ITEST.NE.-1)GO TO 464
      N(I)=9999
432   CARRY=1
      GO TO 522
464   IF(L(I).LE.ITEST)GO TO 498
      N(I)=L(I)-ITEST-1
      GO TO 432
498   N(I)=9999-(ITEST-L(I))
      CARRY=0
522   CONTINUE
      IF(CARRY.NE.0)GO TO 558
      RETURN
558   IF(NN.NE.LIMIT)GO TO 601
582   NN=NN+1
      IF ( NN .GT. 20 ) GO TO 800
      N(NN)=1
      RETURN
601   LIMIT=LIMIT+1
      DO 644 I=LIMIT,NN
      IF(N(I).NE.9999)GO TO 676
      N(I)=0
644   CONTINUE
      GO TO 582
676   N(I)=N(I)+1
      RETURN
800   IND = 1
      RETURN
      END
      SUBROUTINE MULT(J,K,IPROD,N,N1,IND)
      DIMENSION J(20),IPROD(20)
      IPROD(1)=0
      IF(K-1)202,230,280
202   N1=1
      RETURN
230   DO 244 I=1,N
```

```
244     IPROD(I)=J(I)
        N1=N
        RETURN
280     DO 388 I=1,N
        IQ=J(I)/1000
        M=(J(I)-1000*IQ)*K+IPROD(I)
        L=IQ*K
        IQ=L/10
        ITEST=M-(9999-1000*(L-10*IQ))
        IF ( ITEST .LE. 0 ) GO TO 316
        IPROD(I)=ITEST-1
        IQ=IQ+1
        GO TO 350
316     IPROD(I)=ITEST+9999
350     IF(I.EQ.N)GO TO 388
        IPROD(I+1)=IQ
388     CONTINUE
        N1=N
        IF(IQ.EQ.0)RETURN
        N1=N1+1
        IF ( N1 .GT. 20 ) GO TO 500
        IPROD(N1)=IQ
        RETURN
500     IND = 1
        RETURN
        END
        SUBROUTINE SHIFT(IPR,IAS,MP, IND)
        DIMENSION IPR(20)
        IF ( MP .EQ. 0 ) RETURN
        IP=MP
        INTR=0
C       TEST WHETHER THE NUMBER OF PLACES TO BE SHIFTED IS LESS THAN 4
        IF(IP.GE.4)GO TO 290
        KK=1
202     K=10**(4-IP)
        DO 255 I=KK,IAS
        IHIN=IPR(I)/K
        IPR(I)=(IPR(I)-IHIN*K)*10**IP+INTR
        INTR=IHIN
255     CONTINUE
        IF(INTR.EQ.0) RETURN
        IAS=IAS+1
        IF ( IAS .GT. 20 ) GO TO 500
        IPR(IAS)=INTR
        RETURN
290     ID=IP/4
        IF ( IAS+ ID .GT. 20 ) GO TO 500
        DO 311 I=1,IAS
311     IPR(IAS+ID+1-I)=IPR(IAS+1-I)
        IAS=IAS+ID
        DO 355 I=1,ID
```

```
355     IPR(I)=0
C       TEST WHETHER THE NUMBER OF PLACES TO BE SHIFTED IS EXACTLY
C       DIVISIBLE BY 4
        IP=IP-4*ID
        IF(IP.EQ.0)RETURN
        II=ID+1
        GO TO 202
500     IND = 1
        RETURN
        END
        SUBROUTINE INTEG(B,J,D,IDN)
        INTEGER B(20),D(80)
        IDN=0
        ND=4
        DO 566 II=1,J
        IDIV=B(II)
        IF(II-J)442,256,600
202     ND=JJ
        GO TO 442
256     DO 377 JJ=1,3
        IF(IDIV.LT.10**JJ)GO TO 202
377     CONTINUE
442     DO 488 KI=1,ND
        IDN=IDN+1
        D(IDN)=MOD(IDIV,10)
        IF(KI.EQ.ND)GO TO 488
        IDIV=IDIV/10
488     CONTINUE
566     CONTINUE
600     RETURN
        END
        SUBROUTINE PRINT(M)
        IF(M.LT.10)GO TO 1
        IF(M.LT.100)GO TO 2
        IF(M.LT.1000)GO TO 3
        WRITE(1,70)M
70      FORMAT(1X,I4$)
        RETURN
3       WRITE(1,71)M
71      FORMAT(1X,1H0,I3$)
        RETURN
2       WRITE(1,72)M
72      FORMAT(1X,2H00,I2$)
        RETURN
1       WRITE(1,73)M
73      FORMAT(1X,3H000,I1$)
        RETURN
        END
```

6. DIVISION OF A LARGE NUMBER BY ANOTHER

FOR DIVISION ALSO, WE DIVIDE THE DIVIDEND AND THE DIVISOR INTO A NUMBER OF PIECES JUST LIKE ADDITION. LET THE DIVIDEND HAVE I PIECES AND THE DIVISOR HAVE J PIECES. IF I < J, WE STORE THE QUOTIENT AS 0 IN A SINGLE PIECE AND THE REMAINDER BECOMES THE DIVIDEND ITSELF. IF I = J, WE COMPARE THE PIECES OF THE DIVIDEND AND THE DIVISOR STARTING FROM THE LEFTMOST PIECES JUST LIKE SUBTRACTION. IF WE FIND THAT THE DIVIDEND IS LESS THAN THE DIVISOR, WE DO AS ABOVE. IF WE FIND THAT THE DIVIDEND IS EQUAL TO THE DIVISOR, WE STORE THE QUOTIENT AS 1 IN A SINGLE PIECE AND WE SET THE REMAINDER AS 0 ALSO IN A SINGLE PIECE. IF WE FIND THAT THE DIVIDEND IS GREATER THAN THE DIVISOR, WE DIVIDE THE LAST (LEFTMOST) PIECE OF THE DIVIDEND BY THE LAST (LEFTMOST) PIECE OF THE DIVISOR. WE MULTIPLY THE DIVISOR BY THIS QUOTIENT AND COMPARE THE PRODUCT WITH THE DIVIDEND. IF THE DIVIDEND IS NOT LESS THAN THE PRODUCT, THEN WE ARE THROUGH AND WE PRINT THE QUOTIENT AND THE REMAINDER AS USUAL. IF THE DIVIDEND HAPPENS TO BE LESS THAN THE PRODUCT, WE REDUCE THE QUOTIENT BY UNITY AND REPEAT THE PROCESS TILL THE NEW PRODUCT BECOMES LESS THAN OR EQUAL TO THE DIVIDEND. IN THIS WAY WE CALCULATE THE QUOTIENT AND THE REMAINDER.

IF I > J, WE DIGITISE THE RIGHTMOST I-J PIECES OF THE DIVIDEND USING THE SUBROUTINE CALLED INTEG WHICH WE HAVE ALREADY USED IN THE PREVIOUS SECTION. THEN WE COMPARE THE LEFTMOST J PIECES OF THE DIVIDEND WITH THE DIVISOR AND PROCEED IN THE WAY WHICH WE HAVE DESCRIBED ABOVE. AFTER CALCULATING THE PARTIAL QUOTIENT, WE MAKE THE

NEW DIVIDEND GREATER THAN OR EQUAL TO THE DIVISOR BY ATTACHING EXTRA DIGIT(S) FROM THE EXTREME LEFT OF THE REMAINING DIGITS IN THE DIVIDEND USING THE SUBROUTINE CALLED ISOR, AS WE DO IN THE HUMAN DOMAIN. NOW WE AGAIN DIVIDE THE NEW DIVIDEND BY THE DIVISOR AS EXPLAINED BEFORE, WHICH GIVES US 1 MORE DIGIT IN THE QUOTIENT. WE USE THE SUBROUTINES BIGMUL AND BIGSUB FOR MULTIPLICATION AND SUBTRACTION RESPECTIVELY. IF ON ATTACHING 1 SINGLE DIGIT FROM THE GIVEN DIVIDEND, WE FIND THAT THE NUMBER TO BE DIVIDED REMAINS LESS THAN THE DIVISOR, WE ATTACH 0 TO THE DIVISOR JUST AS WE DO IN THE HUMAN DOMAIN. WE REPEAT THE PROCESS TILL ALL THE DIGITS ARE EXHAUSTED. AFTER CALCULATION WE PRINT THE DIVIDEND, DIVISOR, QUOTIENT AND REMAINDER AS USUAL IN SEPARATE LINES USING THE SUBROUTINE PRINT TO AVOID BLANKS IN PLACE OF EMBEDDED ZEROS. THE MAIN PROGRAM FOR DIVISION TOGETHER WITH ALL RELEVANT SUBROUTINES APPEARS ON THE SUCCEEDING PAGES.

//

```
      INTEGER A(20),A1(20),IGIT(80),IQT(20),B(20),PR(20),REM(20)
6     WRITE ( 1 , 34 )
34    FORMAT ( 1X , 'TYPE THE NUMBERS OF PIECES IN THE DIVIDEND',
     11X , 'AND THE DIVISOR IN FORMAT ( 2I2 )' )
154   READ(1,1)I,J
1     FORMAT(2I2)
      IF(I.EQ.0.AND.J.EQ.0)STOP
      IF ( I .GT. 0 .AND. J .GT. 0 ) GO TO 181
      WRITE ( 1 , 89 )
89    FORMAT ( 1X , 'DATA ILLEGAL, PLEASE TYPE AGAIN' )
      GO TO 154
181   IF ( I .LE. 20 .AND. J .LE. 20 ) GO TO 205
      IF ( I .LE. 20 .OR. J .LE. 20 ) GO TO 207
      WRITE ( 1 , 90 )
90    FORMAT ( 1X , 'BOTH NUMBERS TOO LARGE, PLEASE TYPE AGAIN' )
      GO TO 154
207   IF ( I .LE. 20 ) GO TO 305
      WRITE ( 1 , 91 )
91    FORMAT ( 1X , 'DIVIDEND TOO LARGE' )
      GO TO 154
305   WRITE ( 1 , 92 )
92    FORMAT ( 1X , 'DIVISOR TOO LARGE' )
      GO TO 154
205   WRITE ( 1 , 109 )
109   FORMAT ( 1X , 'TYPE THE DIVIDEND IN FORMAT ( 20I4 )' )
      READ ( 1 , 86 ) ( A ( I + 1 - F ) , F = 1 , I )
      WRITE ( 1 , 110 )
110   FORMAT ( 1X , 'TYPE THE DIVISOR IN FORMAT ( 20I4 )' )
      READ ( 1 , 86 ) ( B ( J + 1 - L ) , L = 1 , J )
86    FORMAT ( 20I4 )
      IF(J.GT.1.OR.B(1).GT.0)GO TO 172
      WRITE(1,11)
      GO TO 6
172   IND = 0
      M=1
      IQT(1)=0
      IF(I-J)297,202,414
202   NDI=0
235   MT=J
C     TRANSFER THE LEFTMOST J PIECES FROM THE DIVIDEND TO A NEW ARRAY
      DO 266 II=1,J
266   A1(J+1-II)=A(I+1-II)
294   DO 344 II=1,J
      IF(A1(J+1-II)-B(J+1-II))278,344,485
344   CONTINUE
      MT=1
      A1(1)=0
      IW=1
      GO TO 301
278   IF(IQT(M).EQ.0)GO TO 384
316   IW=0
```

```
301     CALL QUOT(IQT,M,IW,IND)
384     IF(NDI.EQ.0)GO TO 297
        CALL ISOR(A1,MT,IGIT,NDI, IND)
C       COMPARE THE NUMBERS OF PIECES OF THE NEW DIVIDEND AND THE DIVISOR
        IF(MT-J)316,294,600
414     LM=I-J
C       CHECK WHETHER ALL THE DIGITS HAVE BEEN STORED.  IF NOT, STORE THE
C       VACANT PLACES WITH ZEROS
        CALL INTEG(A,LM,IGIT,NDI)
        IC=4*LM
        IF(NDI.EQ.IC) GO TO 235
        NDI=NDI+1
        DO 455 II = NDI , IC
455     IGIT ( II ) = 0
        NDI=IC
        GO TO 235
C       FIND THE TRIAL QUOTIENT
485     IW=A1(J)/B(J)
501     CALL DIVISO(B,J,A1,MT,IW,IND)
        GO TO 301
C       IF BOTH PIECES HAVE 4 DIGITS EACH AND THE LAST PIECE OF THE
C       DIVISOR IS GREATER THAN THE LAST PIECE OF THE DIVIDEND, FIND THE
C       TRIAL DIVISOR
600     IW=(A1(MT)*10+A1(MT-1)/1000)/(B(J)/1000)
        GO TO 501
297     CALL BIGMUL(B,J,IQT,M,PR,L,IND)
        CALL BIGSUB(A,PR,REM,I,L,I,INDEX)
        WRITE(1,17)A(I)
17      FORMAT(1X,I4$)
        IF(I.EQ.1)GO TO 72
        DO 511 I2=2,I
511     CALL PRINT(A(I+1-I2))
72      WRITE(1,213)
213     FORMAT(10X)
        WRITE(1,17)B(J)
        IF(J.EQ.1)GO TO 73
        DO 512 J2=2,J
512     CALL PRINT(B(J+1-J2))
73      WRITE(1,213)
        WRITE(1,17)IQT(M)
        IF(M.EQ.1)GO TO 74
        DO 513 K2=2,M
513     CALL PRINT(IQT(M+1-K2))
74      WRITE(1,213)
        WRITE(1,17)REM(K)
        IF(K.EQ.1)GO TO 75
        DO 514 KK2=2,K
514     CALL PRINT(REM(K+1-KK2))
75      WRITE(1,213)
        GO TO 6
11      FORMAT(1X,'DIVISION BY ZERO IS NOT DEFINED')
```

```
      END
      SUBROUTINE QUOT(IQT,M,IW,IND)
      DIMENSION IQT(20)
      IF(IQT(M).NE.0)GO TO 401
      IQT(1)=IW
      RETURN
401   CALL SHIFT(IQT,M,1,IND)
      IQT(1)=IQT(1)+IW
      RETURN
      END
      SUBROUTINE ISOR(A1,MT,IGIT,NDI,IND)
      INTEGER A1(20),IGIT(80)
      IF(A1(MT).NE.0)GO TO 201
      IF(IGIT(NDI).EQ.0)GO TO 400
      MT=1
      A1(1)=IGIT(NDI)
      GO TO 400
201   CALL SHIFT(A1,MT,1, IND )
      A1(1)=A1(1)+IGIT(NDI)
400   NDI=NDI-1
      RETURN
      END
      SUBROUTINE DIVISO(B,J,A1,MT,IW,IND)
      INTEGER B(20),A1(20),D1(20),C(20),C1(20)
202   C(1)=IW
      CALL BIGMUL(B,J,C,1,D1,NN,IND)
      IF ( IND .EQ. 1 )GO TO 350
      IF(NN-MT)241,208,400
208   DO 211 KK=1,NN
      IF(D1(NN+1-KK)-A1(NN+1-KK))241,211,400
211   CONTINUE
      A1(1)=0
      MT=1
      RETURN
241   CALL BIGSUB(A1,D1,C1,MT,NN,MM,INDEX)
      DO 244 II=1,MM
244   A1(II)=C1(II)
      MT=MM
      RETURN
350   IND=0
400   IW=IW-1
      GO TO 202
      END
      SUBROUTINE BIGMUL(A,I,B,J,C,M,IND)
      INTEGER A(20),IPROD(20),B(20),C(20),D(80),INP(9),ISP(9,20),C1(20)
      IS= - 1
      C ( 1 ) = 0
      M = 1
      DO 222 K = 2 , 9
222   INP ( K ) = 0
```

```
      CALL INTEG(B,J,D,IDN)
      DO 555 MM=1,IDN
      IS = IS + 1
      IF ( D ( MM ) - 1 ) 555 , 386 , 242
242   IF ( INP ( D ( MM ) ) .NE. 0 ) GO TO 320
      CALL MULT(A,D(MM),IPROD,I,N1, IND )
      IF(IND.EQ.1)RETURN
      DO 288 K = 1 , N1
288   ISP ( D ( MM ) , K ) = IPROD ( K )
      INP ( D ( MM ) ) = N1
      GO TO 401
320   N1 = INP ( D ( MM ) )
      DO 355 K = 1 , N1
355   IPROD ( K ) = ISP ( D ( MM ) , K )
      GO TO 401
386   DO 422 K = 1 , I
422   IPROD ( K ) = A ( K )
      N1 = I
401   IF ( N1 .EQ. 1 .AND. IPROD ( N1 ) .EQ. 0 ) GO TO 555
      CALL SHIFT ( IPROD , N1 , IS , IND )
      IF ( IND .EQ. 1 )RETURN
      CALL ADDN(C,IPROD,C1,M,N1,M1,IND )
      IF ( IND .EQ. 1 ) RETURN
      M = M1
      DO 488 MK = 1 , M
488   C ( MK ) = C1 ( MK )
555   CONTINUE
      RETURN
      END
      SUBROUTINE ADDN(L,M,N,LL,MM, NN , IND )
      INTEGER CARRY
      DIMENSION L(20),M(20),N(20)
      CARRY = 0
      IF(LL-MM)300,276,202
202   LIMIT=MM
      IN=MM+1
      DO 244 II=IN,LL
244   N(II)=L(II)
      NN=LL
      GO TO 400
276   LIMIT=MM
      NN=LIMIT
      GO TO 400
300   LIMIT=LL
      IN=LL+1
      DO 344 II=IN,MM
344   N(II)=M(II)
      NN=MM
400   DO 522 I=1,LIMIT
      KTEST=9999-M(I)-CARRY
      IF(L(I).NE.9999.OR.KTEST.NE.-1)GO TO 464
```

```
      N(I)=9999
 432  CARRY=1
      GO TO 522
 464  IF(L(I).LE.KTEST)GO TO 498
      N(I)=L(I)-KTEST-1
      GO TO 432
 498  N(I)=9999-(KTEST-L(I))
      CARRY=0
 522  CONTINUE
      IF(CARRY.NE.0)GO TO 558
      RETURN
 558  IF(NN.NE.LIMIT)GO TO 601
 582  NN=NN+1
      IF ( NN .GT. 20 ) GO TO 800
      N(NN)=1
      RETURN
 601  LIMIT=LIMIT+1
      DO 644 I=LIMIT,NN
      IF(N(I).NE.9999)GO TO 676
      N(I)=0
 644  CONTINUE
      GO TO 582
 676  N(I)=N(I)+1
      RETURN
 800  IND = 1
      RETURN
      END
      SUBROUTINE MULT(J,K,IPROD,N,N1,IND)
      DIMENSION J(20),IPROD(20)
      IPROD(1)=0
      IF(K-1)202,230,280
 202  N1=1
      RETURN
 230  DO 244 I=1,N
 244  IPROD(I)=J(I)
      N1=N
      RETURN
 280  DO 388 I=1,N
      IQ=J(I)/1000
      M=(J(I)-1000*IQ)*K+IPROD(I)
      L=IQ*K
      IQ=L/10
      ITEST=M-(9999-1000*(L-10*IQ))
      IF ( ITEST .LE. 0 ) GO TO 316
      IPROD(I)=ITEST-1
      IQ=IQ+1
      GO TO 350
 316  IPROD(I)=ITEST+9999
 350  IF(I.EQ.N)GO TO 388
      IPROD(I+1)=IQ
```

```
388   CONTINUE
      N1=N
      IF(IO.EQ.0)RETURN
      N1=N1+1
      IF ( N1 .GT. 20 ) GO TO 500
      IPROD(N1)=IO
      RETURN
500   IND = 1
      RETURN
      END
      SUBROUTINE SHIFT(IPR,IAS,MP, IND)
      DIMENSION IPR(20)
      IF ( MP .EQ. 0 ) RETURN
      IP=MP
      INTR=0
      IF(IP.GE.4)GO TO 290
      KK=1
202   K=10**(4-IP)
      DO 255 I=KK,IAS
      IHIN=IPR(I)/K
      IPR(I)=(IPR(I)-IHIN*K)*10**IP+INTR
      INTR=IHIN
255   CONTINUE
      IF(INTR.EQ.0) RETURN
      IAS=IAS+1
      IF ( IAS .GT. 20 ) GO TO 500
      IPR(IAS)=INTR
      RETURN
290   ID=IP/4
      IF ( IAS+ ID .GT. 20 ) GO TO 500
      DO 311 I=1,IAS
311   IPR(IAS+ID+1-I)=IPR(IAS+1-I)
      IAS=IAS+ID
      DO 355 I=1,ID
355   IPR(I)=0
      IP=IP-4*ID
      IF(IP.EQ.0)RETURN
      KK=ID+1
      GO TO 202
500   IND = 1
      RETURN
      END
      SUBROUTINE INTEG(B,J,D,IDN)
      INTEGER B(20),D(80)
      IDN=0
      ND=4
      DO 566 II=1,J
      IDIV=B(II)
      IF(II-J)442,256,600
202   ND=JJ
```

```
      GO TO 442
256   DO 377 JJ=1,3
      IF(IDIV.LT.10**JJ)GO TO 202
377   CONTINUE
442   DO 488 KK=1,ND
      IDN=IDN+1
      D(IDN)=MOD(IDIV,10)
      IF(KK.EQ.ND)GO TO 488
      IDIV=IDIV/10
488   CONTINUE
566   CONTINUE
600   RETURN
      END
      SUBROUTINE BIGSUB(L,M,N,LL,MM,NN,INDEX)
      INTEGER CARRY
      DIMENSION L(20),N1(20),M(20),N(20)
      CARRY=0
      IF(LL-MM)590,681,202
202   INDEX = 0
      DO 244 I=1,MM
244   N1(I)=M(I)
      DO 288 I=1,LL
288   N(I)=L(I)
      NN=LL
320   KK=MM
350   DO 488 I=1,KK
      IDIFF=N(I)-N1(I)
      IF(IDIFF)451,421,386
386   N(I)=IDIFF-CARRY
      CARRY=0
      GO TO 488
421   IF(CARRY.NE.0)GO TO 451
      N(I)=0
      GO TO 488
451   N(I)=IDIFF+1-CARRY+9999
      CARRY=1
488   CONTINUE
      IF(KK.EQ.NN)GO TO 754
      IF(CARRY.NE.0)GO TO 512
      RETURN
512   IF(KK+1.EQ.NN)GO TO 553
      I1=NN-1
      II=KK+1
      DO 533 I=II,I1
      IF(N(I).EQ.0)GO TO 533
      N(I)=N(I)-1
      RETURN
533   N(I)=9999
553   N(NN)=N(NN)-1
      GO TO 754
```

```
590     INDEX=1
        DO 622 I=1,MM
622     N(I)=M(I)
        DO 655 I=1,LL
655     N1(I)=L(I)
        KK=LL
        NN=MM
        GO TO 350
681     DO 711 I=1,MM
        IF(L(MM+1-I)-M(MM+1-I))590,711,202
711     CONTINUE
        INDEX = 0
        NN=1
        N(1)=0
        RETURN
754     IF(NN.EQ.1)RETURN
        MN1=NN-1
        DO 888 I=1,MN1
        IF(N(NN).NE.0) RETURN
888     NN=NN-1
        RETURN
        END
        SUBROUTINE PRINT(M)
        IF(M.LT.10)GO TO 1
        IF(M.LT.100)GO TO 2
        IF(M.LT.1000)GO TO 3
        WRITE(1,70)M
70      FORMAT(1X,I4$)
        RETURN
3       WRITE(1,71)M
71      FORMAT(1X,1H0,I3$)
        RETURN
2       WRITE(1,72)M
72      FORMAT(1X,2H00,I2$)
        RETURN
1       WRITE(1,73)M
73      FORMAT(1X,3H000,I1$)
        RETURN
        END

//
```

7. H.C.F. OF 2 LARGE NUMBERS

THIS PROGRAM IS ONLY A NATURAL EXTENSION OF THE PREVIOUS PROGRAMS WHICH WE HAVE ALREADY DISCUSSED. JUST AS WE DO IN THE HUMAN DOMAIN, WE DIVIDE ONE NUMBER BY ANOTHER USING THE SUBROUTINE BIGDIV. IF THE REMAINDER HAPPENS TO BE ZERO, THE DIVISOR BECOMES THE H.C.F. OTHERWISE WE MOVE THE DIVISOR TO THE DIVIDEND AND THE REMAINDER TO THE DIVISOR, AND REPEAT THE PROCESS TILL THE DIVISION COMES OUT EXACT. THEN WE PRINT THE 2 GIVEN NUMBERS AND ALSO THEIR H.C.F. ON SEPARATE LINES IN THE USUAL FASHION. THIS PROGRAM USES ALL THE SUBROUTINES DISCUSSED IN THIS CHAPTER, NAMELY, BIGDIV, QUOT, ISOR, DIVISO, BIGMUL, INTEG, MULT, SHIFT, ADDN, BIGSUB AND PRINT. THE COMPLETE PROGRAM IS GIVEN ON THE SUCCEEDING PAGES.

//

```
      DIMENSION M(20),N(20),KHCF(20),KQ(20),KREM(20)
83    WRITE(1,11)
11    FORMAT(' TYPE THE NUMBERS OF PIECES IN THE 2 NUMBERS IN',
     11X,'FORMAT(2I2)')
839   READ(1,6)I,J
6     FORMAT(2I2)
      IF(I.EQ.0.AND.J.EQ.0)STOP
      IF(I.GT.0)GO TO 61
9     WRITE(1,12)
12    FORMAT(' DATA ILLEGAL, PLEASE TYPE AGAIN')
      GO TO 839
61    IF(J.GT.0)GO TO 67
      GO TO 9
67    IF(I.LE.20.AND.J.LE.20)GO TO 100
      IF(I.LE.20.OR.J.LE.20)GO TO 89
      WRITE(1,13)
13    FORMAT(' BOTH NUMBERS TOO LARGE, PLEASE TYPE AGAIN')
      GO TO 839
89    IF(I.LE.20)GO TO 891
      WRITE(1,14)
14    FORMAT(' FIRST NUMBER TOO LARGE, PLEASE TYPE AGAIN')
      GO TO 839
891   WRITE(1,15)
15    FORMAT(' SECOND NUMBER TOO LARGE, PLEASE TYPE AGAIN')
      GO TO 839
100   WRITE(1,16)
16    FORMAT(' TYPE THE FIRST NUMBER IN FORMAT(20I4)')
      READ(1,50)(M(I+1-K),K=1,I)
50    FORMAT(20I4)
      WRITE(1,17)
17    FORMAT(' TYPE THE SECOND NUMBER IN FORMAT(20I4)')
      READ(1,50)(N(J+1-L),L=1,J)
      IF(M(I).GT.0.AND.N(J).GT.0)GO TO 1005
      IF(M(I).GT.0.OR.N(J).GT.0)GO TO 1007
      IF(I.GT.1.OR.J.GT.1)GO TO 440
      WRITE(1,531)
531   FORMAT(' HCF IS NOT DEFINED SINCE BOTH NUMBERS ARE ZERO')
      GO TO 83
440   IF(I.GT.1)GO TO 892
27    WRITE(1,20)
      GO TO 100
892   IF(J.GT.1)GO TO 9053
25    WRITE(1,19)
      GO TO 100
9053  WRITE(1,18)
      GO TO 100
1007  IF(M(I).GT.0)GO TO 550
      IF(I.GT.1)GO TO 25
      WRITE(1,1000)M(I)
      WRITE(1,2000)
```

```
      WRITE(1,1000)N(J)
      IF(J.EQ.1)GO TO 1282
      DO 12702 JJ=2,J
12702 CALL PRINT(N(J+1-JJ))
1282  WRITE(1,2000)
      WRITE(1,1000)N(J)
      IF(J.EQ.1)GO TO 128
      DO 127 JJ=2,J
127   CALL PRINT(N(J+1-JJ))
128   WRITE(1,2000)
      GO TO 83
550   IF(J.GT.1)GO TO 27
      WRITE(1,1000)M(I)
      IF(I.EQ.1)GO TO 506
      DO 505 II=2,I
505   CALL PRINT(M(I+1-II))
506   WRITE(1,2000)
      WRITE(1,1000)N(J)
      WRITE(1,2000)
      WRITE(1,1000)M(I)
      IF(I.EQ.1)GO TO 128
      DO 1271 II=2,I
1271  CALL PRINT(M(I+1-II))
      GO TO 128
18    FORMAT(' LAST PIECES OF BOTH NUMBERS ZERO, ABSURD INPUT')
19    FORMAT(' LAST PIECE OF FIRST NUMBER ZERO, ABSURD INPUT')
20    FORMAT(' LAST PIECE OF SECOND NUMBER ZERO, ABSURD INPUT')
1005  WRITE(1,1000)M(I)
      IF(I.EQ.1)GO TO 189
      DO 190 II=2,I
190   CALL PRINT(M(I+1-II))
189   WRITE(1,2000)
      WRITE(1,1000)N(J)
      IF(J.EQ.1)GO TO 1891
      DO 1901 JJ=2,J
1901  CALL PRINT(N(J+1-JJ))
1891  WRITE(1,2000)
1121  CALL BIGDIV(M,N,KQ,KREM,I,J,L,LL)
      IF(LL.EQ.1.AND.KREM(1).EQ.0)GO TO 907
      DO 55 JJ=1,J
55    M(JJ)=N(JJ)
      I=J
      DO 56 LLL=1,LL
56    N(LLL)=KREM(LLL)
      J=LL
      GO TO 1121
907   DO 444 JJJ=1,J
444   KHCF(JJJ)=N(JJJ)
      WRITE(1,1000)KHCF(J)
      IF(J.EQ.1)GO TO 128
      DO 567 JJJJ=2,J
```

```
567     CALL PRINT(KHCF(J+1-JJJJ))
        GO TO 128
1000    FORMAT(1X,I4$)
2000    FORMAT(50X)
        END
        SUBROUTINE BIGDIV(A,B,IQT,REM,I,J,M,K)
        INTEGER A(20),A1(20),IGIT(80),IQT(20),B(20),PR(20),REM(20)
172     IND = 0
        M=1
        IQT(1)=0
        IF(I-J)297,202,414
202     NDI=0
235     MT=J
        DO 266 II=1,J
266     A1(J+1-II)=A(I+1-II)
294     DO 344 II=1,J
        IF(A1(J+1-II)-B(J+1-II))278,344,485
344     CONTINUE
        MT=1
        A1(1)=0
        IW=1
        GO TO 301
278     IF(IQT(M).EQ.0)GO TO 384
316     IW=0
301     CALL QUOT(IQT,M,IW,IND)
384     IF(NDI.EQ.0)GO TO 297
        CALL ISOR(A1,MT,IGIT,NDI, IND)
        IF(MT-J)316,294,600
414     LM=I-J
        CALL INTEG(A,LM,IGIT,NDI)
        IC = 4 * LM
        IF ( NDI .EQ. IC ) GO TO 235
        NDI=NDI+1
        DO 455 II = NDI , IC
455     IGIT ( II ) = 0
        NDI = IC
        GO TO 235
485     IW=A1(J)/B(J)
501     CALL DIVISO(B,J,A1,MT,IW,IND)
        GO TO 301
600     IW=(A1(MT)*10+A1(MT-1)/1000)/(B(J)/1000)
        GO TO 501
297     CALL BIGMUL(B,J,IQT,M,PR,L,IND)
        CALL BIGSUB(A,PR,REM,I,L,K,INDEX)
        RETURN
        END
        SUBROUTINE QUOT(IQT,M,IW,IND)
        DIMENSION IQT(20)
        IF(IQT(M).NE.0)GO TO 401
        IQT(1)=IW
        RETURN
```

```
401     CALL SHIFT(IQT,M,1,IND)
        IQT(1)=IQT(1)+IW
        RETURN
        END
        SUBROUTINE ISOR(A1,MT,IGIT,NDI,IND)
        INTEGER A1(20),IGIT(80)
        IF(A1(MT).NE.0)GO TO 201
        IF(IGIT(NDI).EQ.0)GO TO 400
        MT=1
        A1(1)=IGIT(NDI)
        GO TO 400
201     CALL SHIFT(A1,MT,1, IND )
        A1(1)=A1(1)+IGIT(NDI)
400     NDI=NDI-1
        RETURN
        END
        SUBROUTINE DIVISO(B,J,A1,MT,IW,IND)
        INTEGER B(20),A1(20),D1(20),C(20),C1(20)
202     C(1)=IW
        CALL BIGMUL(B,J,C,1,D1,NN,IND)
        IF(IND.EQ.1)GO TO 350
        IF(NN-MT)241,208,400
208     DO 211 KK=1,NN
        IF(D1(NN+1-KK)-A1(NN+1-KK))241,211,400
211     CONTINUE
        A1(1)=0
        MT=1
        RETURN
241     CALL BIGSUB(A1,D1,C1,MT,NN,MM,INDEX)
        DO 244 II=1,MM
244     A1(II)=C1(II)
        MT=MM
        RETURN
350     IND=0
400     IW=IW-1
        GO TO 202
        END
        SUBROUTINE BIGMUL(A,I,B,J,C,M,IND)
        INTEGER A(20),IPROD(20),B(20),C(20),D(80),INP(9),ISP(9,20),C1(20)
        IS= - 1
        C ( 1 ) = 0
        M = 1
        DO 222 K = 2 , 9
222     INP ( K ) = 0
        CALL INTEG(B,J,D,IDN)
        DO 555 MM=1,IDN
        IS = IS + 1
        IF ( D ( MM ) - 1 ) 555 , 386 , 242
242     IF ( INP ( D ( MM ) ) .NE. 0 ) GO TO 320
        CALL MULT(A,D(MM),IPROD,I,N1, IND )
```

```
      IF(IND.EQ.1)RETURN
      DO 288 K = 1 , N1
288   ISP ( D ( MM ) , K ) = IPROD ( K )
      INP ( D ( MM ) ) = N1
      GO TO 401
320   N1 = INP ( D ( MM ) )
      DO 355 K = 1 , N1
355   IPROD ( K ) = ISP ( D ( MM ) , K )
      GO TO 401
386   DO 422 K = 1 , I
422   IPROD ( K ) = A ( K )
      N1 = I
401   IF ( N1 .EQ. 1 .AND. IPROD ( N1 ) .EQ. 0 ) GO TO 555
      CALL SHIFT ( IPROD , N1 , IS , IND )
      IF ( IND .EQ. 1 )RETURN
      CALL ADDN(C,IPROD,C1,M,N1,M1,IND )
      IF ( IND .EQ. 1 ) RETURN
      M = M1
      DO 488 MK = 1 , M
488   C ( MK ) = C1 ( MK )
555   CONTINUE
      RETURN
      END
      SUBROUTINE ADDN(L,M,N,LL,MM, NN , IND )
      INTEGER CARRY
      DIMENSION L(20),M(20),N(20)
      CARRY = 0
      IF(LL-MM)300,276,202
202   LIMIT=MM
      IN=MM+1
      DO 244 II=IN,LL
244   N(II)=L(II)
      NN=LL
      GO TO 400
276   LIMIT=MM
      NN=LIMIT
      GO TO 400
300   LIMIT=LL
      IN=LL+1
      DO 344 II=IN,MM
344   N(II)=M(II)
      NN=MM
400   DO 522 I=1,LIMIT
      KTEST=9999-M(I)-CARRY
      IF(L(I).NE.9999.OR.KTEST.NE.-1)GO TO 464
      N(I)=9999
432   CARRY=1
      GO TO 522
464   IF(L(I).LE.KTEST)GO TO 498
      N(I)=L(I)-KTEST-1
```

```
      GO TO 432
498   N(I)=9999-(KTEST-L(I))
      CARRY=0
522   CONTINUE
      IF(CARRY.NE.0)GO TO 558
      RETURN
558   IF(NN.NE.LIMIT)GO TO 601
582   NN=NN+1
      IF ( NN .GT. 20 ) GO TO 800
      N(NN)=1
      RETURN
601   LIMIT=LIMIT+1
      DO 644 I=LIMIT,NN
      IF(N(I).NE.9999)GO TO 676
      N(I)=0
644   CONTINUE
      GO TO 582
676   N(I)=N(I)+1
      RETURN
800   IND = 1
      RETURN
      END
      SUBROUTINE MULT(J,K,IPROD,N,N1,IND)
      DIMENSION J(20),IPROD(20)
      IPROD(1)=0
      IF(K-1)202,230,280
202   N1=1
      RETURN
230   DO 244 I=1,N
244   IPROD(I)=J(I)
      N1=N
      RETURN
280   DO 388 I=1,N
      IQ=J(I)/1000
      M=(J(I)-1000*IQ)*K+IPROD(I)
      L=IQ*K
      IQ=L/10
      ITEST=M-(9999-1000*(L-10*IQ))
      IF ( ITEST .LE. 0 ) GO TO 316
      IPROD(I)=ITEST-1
      IQ=IQ+1
      GO TO 350
316   IPROD(I)=ITEST+9999
350   IF(I .EQ.N)GO TO 388
      IPROD(I+1)=IQ
388   CONTINUE
      N1=N
      IF(IQ.EQ.0)RETURN
      N1=N1+1
      IF ( N1 .GT. 20 ) GO TO 500
      IPROD(N1)=IQ
```

```
      RETURN
500   IND = 1
      RETURN
      END
      SUBROUTINE SHIFT(IPR,IAS,MP, IND)
      DIMENSION IPR(20)
      IF ( MP .EQ. 0 ) RETURN
      IP=MP
      INTR=0
      IF(IP.GE.4)GO TO 290
      KK=1
202   K=10**(4-IP)
      DO 255 I=KK,IAS
      IHIN=IPR(I)/K
      IPR(I)=(IPR(I)-IHIN*K)*10**IP+INTR
      INTR=IHIN
255   CONTINUE
      IF(INTR.EQ.0) RETURN
      IAS=IAS+1
      IF ( IAS .GT. 20 ) GO TO 500
      IPR(IAS)=INTR
      RETURN
290   ID=IP/4
      IF ( IAS+ ID .GT. 20 ) GO TO 500
      DO 311 I=1,IAS
311   IPR(IAS+ID+1-I)=IPR(IAS+1-I)
      IAS=IAS+ID
      DO 355 I=1,ID
355   IPR(I)=0
      IP=IP-4*ID
      IF(IP.EQ.0)RETURN
      KK=ID+1
      GO TO 202
500   IND = 1
      RETURN
      END
      SUBROUTINE INTEG(B,J,D,IDN)
      INTEGER B(20),D(80)
      IDN=0
      ND=4
      DO 566 II=1,J
      IDIV=B(II)
      IF(II-J)442,256,600
202   ND=JJ
      GO TO 442
256   DO 377 JJ=1,3
      IF(IDIV.LT.10**JJ)GO TO 202
377   CONTINUE
442   DO 488 KK=1,ND
      IDN=IDN+1
```

```
      D(IDN)=MOD(IDIV,10)
      IF(KK.EQ.ND)GO TO 488
      IDIV=IDIV/10
488   CONTINUE
566   CONTINUE
600   RETURN
      END
      SUBROUTINE BIGSUB(L,M,N,LL,MM,NN,INDEX)
      INTEGER CARRY
      DIMENSION L(20),N1(20),M(20),N(20)
      CARRY=0
      IF(LL-MM)590,681,202
202   INDEX = 0
      DO 244 I=1,MM
244   N1(I)=M(I)
      DO 288 I=1,LL
288   N(I)=L(I)
      NN=LL
320   KK=MM
350   DO 488 I=1,KK
      IDIFF=N(I)-N1(I)
      IF(IDIFF)451,421,386
386   N(I)=IDIFF-CARRY
      CARRY=0
      GO TO 488
421   IF(CARRY.NE.0)GO TO 451
      N(I)=0
      GO TO 488
451   N(I)=IDIFF+1-CARRY+9999
      CARRY=1
488   CONTINUE
      IF(KK.EQ.NN)GO TO 754
      IF(CARRY.NE.0)GO TO 512
      RETURN
512   IF(KK+1.EQ.NN)GO TO 553
      I1=NN-1
      II=KK+1
      DO 533 I=II,I1
      IF(N(I).EQ.0)GO TO 533
      N(I)=N(I)-1
      RETURN
533   N(I)=9999
553   N(NN)=N(NN)-1
      GO TO 754
590   INDEX=1
      DO 622 I=1,MM
622   N(I)=M(I)
      DO 655 I=1,LL
655   N1(I)=L(I)
      KK=LL
      NN=MM
```

```
      GO TO 350
681   DO 711 I=1,MM
      IF(L(MM+1-I)-M(MM+1-I))590,711,202
711   CONTINUE
      INDEX = 0
      NN=1
      N(1)=0
      RETURN
754   IF(NN.EQ.1)RETURN
      MN1=NN-1
      DO 888 I=1,MN1
      IF(N(NN).NE.0) RETURN
      NN=NN-1
888   CONTINUE
      RETURN
      END
      SUBROUTINE PRINT(M)
      IF(M.LT.10)GO TO 1
      IF(M.LT.100)GO TO 2
      IF(M.LT.1000)GO TO 3
      WRITE(1,70)M
70    FORMAT(1X,I4$)
      RETURN
3     WRITE(1,71)M
71    FORMAT(1X,1H0,I3$)
      RETURN
2     WRITE(1,72)M
72    FORMAT(1X,2H00,I2$)
      RETURN
1     WRITE(1,73)M
73    FORMAT(1X,3H000,I1$)
      RETURN
      END
```

CHAPTER 2

SOME PROGRAMS REGARDING POLYNOMIALS

1. INTRODUCTION

IN THIS CHAPTER WE SHALL DEVELOP SOME PROGRAMS REGARDING THE MANIPULATION OF POLYNOMIALS IN COMPUTERS. A COMPUTER CANNOT READ OR PRINT A POLYNOMIAL AS SUCH, BUT IF THE POLYNOMIAL IS IN ONE VARIABLE ONLY, FOR EXAMPLE,-

$$A_0 X^N + A_1 X^{N-1} + A_2 X^{N-2} + \ldots + A_{N-1} X + A_N$$

WHERE N IS A NATURAL NUMBER AND ALL COEFFICIENTS ARE REAL NUMBERS, WE CAN STORE THE COEFFICIENTS OF THIS POLYNOMIAL IN A REAL 1-DIMENSIONAL ARRAY. IF THE POLYNOMIAL IS IN MORE THAN ONE VARIABLE, WE REQUIRE A HIGHER-DIMENSIONAL ARRAY TO STORE ITS CO-EFFICIENTS. HOWEVER, IN THIS CHAPTER WE SHALL DEAL WITH POLYNO-MIALS IN ONE VARIABLE ONLY. FURTHER IN SOME OF THE PROBLEMS WE SHALL ASSUME THAT ALL THE COEFFICIENTS ARE INTEGERS AND THE LEAD-ING COEFFICIENT IS NON-ZERO. IF THE POLYNOMIAL HAPPENS TO BE MONIC, WE SHALL STORE ONLY THE N COEFFICIENTS IN AN INTEGER ARRAY AND NOT IN A REAL ARRAY. IF THE POLYNOMIAL IS NOT MONIC, WE SHALL STORE ALL THE COEFFICIENTS (INCLUDING THE LEADING COEFFICIENT) IN AN INTEGER ARRAY. BUT IN THIS CASE WE SHALL CALL THE LEADING CO-EFFICIENT AS A(1), THE NEXT COEFFICIENT AS A(2), ETC. IN OTHER PROBLEMS WE SHALL ASSUME THAT ALL COEFFICIENTS ARE RATIONAL NUMBERS AND WE SHALL STORE ALL THE NUMERATORS OF THE COEFFICIENTS IN ONE INTEGER ARRAY AND ALL THE DENOMINATORS IN ANOTHER INTEGER ARRAY.

//

IN SUCH CASES WE SHALL ASSUME THAT THE NUMERATOR OF THE LEADING COEFFICIENT IS NON-ZERO AND ALL DENOMINATORS ARE ALSO NON-ZERO. HOWEVER, WE SHALL NOT ASSUME THAT ALL THE COEFFICIENTS ARE WRITTEN IN THEIR LOWEST TERMS. IN OTHER WORDS, THE H.C.F. OF ANY ENTRY IN THE NUMERATOR ARRAY AND THE CORRESPONDING ENTRY IN THE DENOMINATOR ARRAY NEED NOT BE UNITY. ALSO WE DO NOT ASSUME THAT ALL THE DENOMINATORS ARE POSITIVE. IN CASE THE LEADING ENTRY IN THE NUMERATOR ARRAY HAPPENS TO BE 0, OR ANY ENTRY IN THE DENOMINATOR ARRAY IS 0, OUR PROGRAM WILL PRINT OUT AN ERROR MESSAGE AND ASK FOR FRESH INPUT.

WE OBSERVE THAT THE ENTRIES OF THE VARIOUS ARRAYS INCREASE RAPIDLY AS THE CALCULATION PROCEEDS, EVEN IF WE DIVIDE BOTH NUMERATOR AND DENOMINATOR BY THEIR H.C.F. AFTER EVERY PIECE OF CALCULATION. THE ENTRIES SOON BECOME GREATER THAN 32767 WHICH IS THE MAXIMUM LIMIT FOR AN INTEGER VARIABLE IN OUR COMPUTER. SO WE SHALL HAVE TO USE THE SUBROUTINE SUBPROGRAMS DEVELOPED IN CHAPTER 1. IN THE PROGRAMS DEVELOPED IN THIS CHAPTER, HOWEVER, WE HAVE NOT WRITTEN THIS EXPLICITLY, BUT IF SOMEONE DESIRES, HE CAN VERY EASILY DO SO.

2. PROGRAM FOR RE-CONSTRUCTION OF A POLYNOMIAL FROM ITS ZEROS

THIS PROGRAM IS ALREADY GIVEN IN A BOOK BY E.V.KRISHNAMURTHY AND S.K.SEN. BUT THAT PROGRAM IS VERY LENGTHY AND COMPLICATED. THE AUTHORS FIRST FIND ALL THE COMBINATIONS OF M NATURAL NUMBERS TAKEN N AT A TIME, WHERE N = 1,2,3,...,M. EVEN THAT SUBROUTINE WHICH CALCULATES THESE COMBINATIONS IS VERY LENGTHY AND DOES ITS

JOB IN A VERY ROUNDABOUT WAY. IT OFTEN RE-DEFINES SOME VARIABLES TO THEIR PREVIOUS VALUES, WHICH IS QUITE UNNECESSARY. THE AUTHORS HAVE SAID IN THEIR BOOK, 'THE PROGRAM IS USEFUL FOR A POLYNOMIAL OF SMALL DEGREE, SAY, 10. FOR A POLYNOMIAL OF LARGE DEGREE, IT INVOLVES TOO MANY MULTIPLICATIONS AND ADDITIONS, THUS INTRODUCING APPRECIABLE ERRORS. WE CAN HOWEVER USE A ROUTINE FOR MULTIPLICATION OF 2 POLYNOMIALS OF DEGREES M AND N (KNUTH 1968) TO RE-CONSTRUCT A POLYNOMIAL FROM A GIVEN SET OF ZEROS (ALSO SEE BERZTISS 1971). A VERY SIMPLE AND STRAIGHTFORWARD TECHNIQUE IS TO WRITE A ROUTINE FOR MULTIPLICATION OF AN MTH DEGREE POLYNOMIAL (M .= 1) BY A LINEAR FACTOR. USING THIS ROUTINE, WE CAN EASILY RE-CONSTRUCT A NORMALISED POLYNOMIAL FROM ITS GIVEN ZEROS.' IN OUR PROGRAM WE HAVE USED THIS TECHNIQUE ONLY, BUT WE HAVE RESTRICTED IT TO INTEGER ZEROS AND RATIONAL ZEROS ONLY.

WE TAKE THE FOLLOWING 2 CASES:-

1. ALL ZEROS ARE INTEGERS.
2. ALL ZEROS ARE RATIONAL, THE H.C.F. OF THE NUMERATORS AND THE CORRESPONDING DENOMINATORS NOT BEING NECESSARILY UNITY, BUT ALL DENOMINATORS BEING NON-ZERO.

THE PROGRAM, IN CASE ALL ZEROS HAPPEN TO BE INTEGERS, IS GIVEN ON THE SUCCEEDING PAGES. IN THIS CASE THE POLYNOMIAL GENERATED IS MONIC. SO WE HAVE NOT PRINTED THE LEADING COEFFICIENT, WHICH IS NECESSARILY UNITY. WE HAVE PRINTED ONLY THE SECOND AND LATER CO-EFFICIENTS OF THE POLYNOMIAL.

```
      DIMENSION ITZ ( 25 ) , ICOF ( 25 )
      WRITE ( 1 , 7 )
7     FORMAT ( 1X , 'THIS PROGRAM RE-CONSTRUCTS A MONIC POLYNOMIAL'
     1 1),'FROM ITS ZEROS, ASSUMING THAT ALL'/' OF THEM ARE INTEGERS'
79    WRITE ( 1 , 107 )
107   FORMAT ( 1X , 'TYPE THE NUMBER OF ZEROS IN FORMAT ( I2 ) '
     1 / 1X , 'FOR STOPPING PLEASE TYPE : 0' )
791   READ ( 1 , 127 ) NZ
127   FORMAT ( I2 )
      IF ( NZ ) 149 , 499 , 176
149   WRITE ( 1 , 157 )
157   FORMAT ( 1X , 'DATA ILLEGAL , KINDLY TYPE AGAIN' )
      GO TO 791
C     CHECK WHETHER THE GIVEN NUMBER OF ZEROS WILL CROSS THE DIMENSION
176   IF ( NZ .GT. 25 ) GO TO 459
      WRITE ( 1 , 207 )
207   FORMAT ( 1X , 'TYPE ALL ZEROS IN FORMAT ( 13I6 )' )
      READ ( 1 , 247 ) ( ITZ ( K ) , K = 1 , NZ )
247   FORMAT ( 13I6 )
C     FORM THE FIRST LINEAR POLYNOMIAL FROM THE FIRST ZERO
      ICOF ( 1 ) = - ITZ ( 1 )
      M = 1
      IF ( NZ .EQ. 1 ) GO TO 401
      DO 311 I = 2 , NZ
C     MULTIPLY THE CURRENT POLYNOMIAL BY A LINEAR POLYNOMIAL
311   CALL LIN ( ICOF , M , ITZ ( I ) )
      WRITE ( 1 , 570 )
570   FORMAT ( 1X , 'THE ZEROS OF THE POLYNOMIAL ARE GIVEN BELOW:-' )
401   WRITE ( 1 , 467 ) ( ITZ ( K ) , K = 1 , NZ )
      WRITE ( 1 , 700 )
700   FORMAT ( 1X , 'THE REQUIRED POLYNOMIAL IS MONIC; SO ITS FIRST',
     1 1X , 'COEFFICIENT IS UNITY' / ' ITS OTHER COEFFICIENTS,',
     1 1X , 'STARTING FROM THE SECOND, ARE GIVEN BELOW:-' )
      WRITE ( 1 , 467 ) ( ICOF ( K ) , K = 1 , M )
467   FORMAT ( 1X , 11 ( I6 , 1X ) )
      GO TO 79
459   WRITE ( 1 , 487 )
487   FORMAT ( 1X , 'THE GIVEN NUMBER WILL CROSS THE DIMENSION,'
     1 , 1X , 'CHANGE DIMENSION' )
499   STOP
      END
      SUBROUTINE LIN ( ICOF , M , MLZ )
      DIMENSION ICOF ( 25 )
      M = M + 1
      ICOF ( M ) = - MLZ * ICOF ( M - 1 )
      IF ( M .EQ. 2 ) GO TO 400
      MN2 = M -2
      DO 122  K = 1 , MN2
122   ICOF ( M - K ) = ICOF ( M - K ) - MLZ * ICOF ( M - K - 1 )
400   ICOF ( 1 ) = ICOF ( 1 ) - MLZ
      RETURN
      END
```

//

NOW WE GIVE THE ALGORITHM FOR RE-CONSTRUCTION OF A POLYNOMIAL FROM ITS RATIONAL ZEROS. FOR ALL SUCH ZEROS FOR WHICH THE NUMERATOR IS 0, THE PROGRAM WILL SET THE DENOMINATOR AS UNITY. IF THE DENOMINATOR OF ANY ZERO IS NEGATIVE, IT WILL CHANGE THE SIGNS OF BOTH NUMERATOR AND DENOMINATOR. THEN THE PROGRAM WILL CALCULATE THE H.C.F. OF BOTH NUMERATOR AND DENOMINATOR IN THE CASE OF EVERY ZERO OF THE POLYNOMIAL WHICH IS NON-ZERO AND DIVIDE BOTH OF THEM BY THIS H.C.F. NOW WE FORM A LINEAR POLYNOMIAL WITH THE GIVEN FIRST RATIONAL ZERO WHICH WILL BE OF THE FORM

$$A_1 X - B_1$$

WHERE B_1/A_1 IS THE FIRST RATIONAL ZERO OF THE REQUIRED POLYNOMIAL.

IF THERE IS NO OTHER ZERO, WE ARE THROUGH. OTHERWISE WE FORM ANOTHER FIRST DEGREE POLYNOMIAL WITH THE SECOND RATIONAL ZERO. WE MULTIPLY THE PREVIOUS POLYNOMIAL BY THE NEW POLYNOMIAL AND CONTINUE THE PROCESS OF MULTIPLICATION TILL ALL THE ZEROS ARE EXHAUSTED. IN THE END WE SHALL GET A POLYNOMIAL HAVING N+1 COEFFICIENTS WHERE N IS THE DEGREE OF THE POLYNOMIAL, THE LEADING COEFFICIENT BEING POSITIVE AND ALL COEFFICIENTS BEING INTEGERS. WE NEED NOT FIND THE H.C.F. OF ALL COEFFICIENTS NOW, BECAUSE WE HAVE TAKEN ALL ZEROS IN THEIR LOWEST TERMS. IT FOLLOWS THAT THE H.C.F. OF ALL COEFFICIENTS WILL CERTAINLY BE UNITY. THE PROGRAM FOR THIS PROCEDURE, TOGETHER WITH THE RELEVANT SUBROUTINE FOR MULTIPLICATION OF A POLYNOMIAL BY A LINEAR POLYNOMIAL, IS GIVEN ON THE SUCCEEDING PAGES.

```
//
      DIMENSION NUM ( 25 ) , IDEN ( 25 ) , ICOF ( 26 )
      INTEGER HCF
      WRITE ( 1 , 7 )
7     FORMAT ( 1X , 'THIS PROGRAM RE-CONSTRUCTS A POLYNOMIAL WITH',
     1 1X , 'INTEGER COEFFICIENTS' / 1X , 'FROM ITS RATIONAL ZEROS' )
79    WRITE ( 1 , 107 )
107   FORMAT ( 1X , 'TYPE THE NUMBER OF ZEROS IN FORMAT ( I2 )' / 1X ,
     1 'FOR STOPPING PLEASE TYPE : 0' )
792   READ ( 1 , 127 ) NZ
127   FORMAT ( I2 )
      IF ( NZ ) 179 , 799 , 238
179   WRITE ( 1 , 207 )
207   FORMAT ( 1X , 'DATA ILLEGAL, KINDLY TYPE  AGAIN' )
      GO TO 792
238   IF ( NZ .GT. 25 ) GO TO 749
      WRITE ( 1 , 267 )
267   FORMAT ( 1X , 'TYPE ALL NUMERATORS OF THE ZEROS IN',
     1 1X , 'FORMAT ( 13I6 )' )
      READ ( 1 , 387 ) ( NUM ( K ) , K = 1 , NZ )
309   WRITE ( 1 , 347 )
347   FORMAT ( 1X , 'TYPE ALL DENOMINATORS OF THE ZEROS IN',
     1 1X , 'FORMAT ( 13I6 )' )
4000  READ ( 1 , 387 ) ( IDEN ( K ) , K = 1 , NZ )
387   FORMAT ( 13I6 )
      DO 411 K = 1 , NZ
      IF ( IDEN ( K ) .EQ. 0 ) GO TO 579
411   CONTINUE
C     STANDARDISE THE DATA
      DO 566  K = 1 , NZ
      ICN = NUM ( K )
      ICD = IDEN ( K )
      IF ( ICN .NE. 0 ) GO TO 446
      ICD = 1
      GO TO 495
446   IF ( ICD .GT. 0 ) GO TO 474
      ICD = - ICD
      ICN = - ICN
474   IHCF = HCF ( ICN , ICD )
      IF ( IHCF .EQ. 1 ) GO TO 495
      ICN = ICN / IHCF
      ICD = ICD / IHCF
495   IF ( K .NE. 1 ) GO TO 501
C     FORM A LINEAR POLYNOMIAL WITH THE FIRST ZERO
      ICOF ( 1 ) = ICD
      ICOF ( 2 ) = - ICN
      M = 2
      GO TO 566
501   CALL LIN ( ICOF , M , ICN , ICD )
566   CONTINUE
      WRITE ( 1 , 567)
```

```
567     FORMAT ( 1X , 'THE NUMERATORS OF THE ZEROS OF THE REQUIRED',
     1 1X , 'POLYNOMIAL ARE GIVEN BELOW:-' )
        WRITE ( 1 , 587 ) ( NUM ( K ) , K = 1 , NZ )
        WRITE ( 1 , 908 )
908     FORMAT ( 1X , 'THE DENOMINATORS OF THE ZEROS OF THE REQUIRED',
     1 1X , 'POLYNOMIAL ARE GIVEN BELOW:-' )
        WRITE ( 1 , 587 ) ( IDEN ( K ) , K = 1 , NZ )
        WRITE ( 1 , 666 )
666     FORMAT ( 1X , 'THE COEFFICIENTS OF THE REQUIRED POLYNOMIAL' ,
     1 1X , 'ARE GIVEN BELOW:-' )
        WRITE ( 1 , 587 ) ( ICOF ( K ) , K = 1 , M )
587     FORMAT ( 1X , 11 ( I6 , 1X ) )
        GO TO 79
579     WRITE ( 1 , 627 )
        GO TO 4000
627     FORMAT ( 1X , 'NO DENOMINATOR CAN BE ZERO, KINDLY TYPE' , 1X ,
     1 'ALL THE DENOMINATORS AGAIN' )
749     WRITE ( 1 , 787)
787     FORMAT ( 1X , 'THE GIVEN NUMBER WILL CROSS THE DIMENSION,' , 1X,
     1 'CHANGE DIMENSION' )
799     STOP
        END
        SUBROUTINE LIN ( ICOF , M , MLNZ , MLDZ )
        DIMENSION ICOF ( 26 )
        M = M + 1
        ICOF ( M ) = - MLNZ * ICOF ( M - 1 )
        M2 = M - 2
        DO 122 K = 1 , M2
122     ICOF ( M - K ) = MLDZ * ICOF ( M - K ) - MLNZ *ICOF( M - K - 1 )
        ICOF ( 1 ) = MLDZ * ICOF ( 1 )
        RETURN
        END
        INTEGER FUNCTION HCF ( I , J )
        INTEGER REM
        IF ( I .NE. 0 ) GO TO 85
        HCF = IABS ( J )
        RETURN
85      IF ( J .NE. 0 ) GO TO 153
        HCF = IABS ( I )
        RETURN
153     KOPY1 = IABS ( I )
        KOPY2 = IABS ( J )
900     REM = MOD ( KOPY1 , KOPY2 )
        IF ( REM .NE. 0 ) GO TO 815
        HCF = KOPY2
        RETURN
815     KOPY1 = KOPY2
        KOPY2 = REM
        GO TO 900
        END
```

//
3. PROGRAM FOR CALCULATING ALL INTEGER ZEROS OF MONIC POLYNOMIAL

WE ASSUME THAT ALL COEFFICIENTS OF THE POLYNOMIAL ARE INTEGERS. THIS IS A MUCH MORE DIFFICULT PROBLEM. WE HAVE SOLVED IT BY THE USE OF THE REMAINDER THEOREM. WE CONSIDER THE LAST NON-ZERO COEFFICIENT OF THE POLYNOMIAL, IF ANY. IF THERE IS NO SUCH COEFFICIENT, THE POLYNOMIAL BECOMES JUST X^N. WE PRINT ALL ITS ZEROS AS 0. IF THE LAST NON-ZERO COEFFICIENT IS A(1), WE PRINT (N-1) ZEROS AS 0 AND THE NTH ZERO AS -A(1). IN ALL OTHER CASES WE FIND ALL POSSIBLE FACTORS OF THE LAST NON-ZERO COEFFICIENT BY USING A SUBROUTINE CALLED FAC. WE FIND THE VALUE OF THE POLYNOMIAL FOR THE FIRST FACTOR. IF THE VALUE OF THE POLYNOMIAL IS 0, THEN THE POLYNOMIAL WILL HAVE THAT FACTOR AS ITS ZERO. WE DIVIDE THE POLYNOMIAL BY X-F, WHERE F IS THAT FACTOR AND CHECK THE DEGREE OF THE NEW POLYNOMIAL. IF ITS DEGREE IS UNITY, THEN THE CONSTANT TERM OF THE NEW POLYNOMIAL WITH ITS SIGN CHANGED WILL BE THE LAST ZERO OF THE ORIGINAL POLYNOMIAL. IF THE FIRST FACTOR DOES NOT SATISFY THE ORIGINAL POLYNOMIAL, WE CHECK THE NEGATIVE OF THE FIRST FACTOR. THEN WE CHECK THE SECOND FACTOR AND SO ON. AS SOON AS WE GET ANY FACTOR WHICH SATISFIES THE POLYNOMIAL, WE IMMEDIATELY DIVIDE THE POLYNOMIAL AS BEFORE AND CONTINUE FURTHER. WE TAKE PARTICULAR PRECAUTIONS THAT IF A FACTOR DOES NOT SATISFY THE POLYNOMIAL, WE DO NOT TEST THAT FACTOR IN THE NEW POLYNOMIALS WHICH ARE OBTAINED AFTER DIVIDING THE ORIGINAL POLYNOMIAL BY LINEAR POLYNOMIALS. WE ONLY CHECK THE SUBSEQUENT FACTORS. IN THE PROCESS WE FIND ALL

//

FACTORS OF THE LAST NON-ZERO COEFFICIENT OF THE ORIGINAL POLYNOMIAL ONLY ONCE. WE DO NOT CALCULATE THE FACTORS OF THE LAST NON-ZERO COEFFICIENTS OF OTHER POLYNOMIALS OF SMALLER DEGRESS WHICH ARE OBTAINED DURING THE PROCESS OF DIVISION. CONTINUING TILL THE END, THERE ARE 3 POSSIBILITIES.

1. ALL ZEROS HAVE BEEN OBTAINED. WE PRINT THEM TOGETHER WITH AN APPROPRIATE MESSAGE. IN THIS CASE THE LAST ZERO WILL ALWAYS BE OBTAINED FROM A LINEAR POLYNOMIAL.
2. SOME OF THE ZEROS HAVE BEEN OBTAINED BUT NOT ALL. IN THIS CASE THE LAST POLYNOMIAL WILL BE OF DEGREE AT LEAST 2 AND NONE OF THE REMAINING FACTORS, IF ANY, OF THE LAST CO-EFFICIENT OF THE ORIGINAL POLYNOMIAL WILL SATISFY THE LAST POLYNOMIAL. SO WE PRINT THE ZEROS OBTAINED TOGETHER WITH ANOTHER APPROPRIATE MESSAGE.
3. NONE OF THE ZEROS ARE INTEGERS. IN THIS CASE WE ONLY PRINT AN APPROPRIATE MESSAGE.

THE SUBROUTINE CALLED FAC HAS BEEN WRITTEN BY MY TEACHER. IT GIVES ALL FACTORS OF A GIVEN NATURAL NUMBER. FIRST WE CALCULATE THE INTEGER PART OF THE SQUARE ROOT OF THE GIVEN NUMBER BY THE USUAL ALGORITHM GIVEN IN ARITHMETIC BOOKS. WE DO NOT DO THIS BY USING THE SQRT FUNCTION OR WRITING A**0.5. IN THIS WAY WE ENSURE THAT ONLY INTEGER ARITHMETIC IS USED AND NOT REAL ARITHMETIC. THUS THE LIBRARY FUNCTIONS ALOG AND EXP ARE ALSO NEVER INVOKED. WE SET A LIMIT EQUAL TO THIS NUMBER. THEN WE PUT DIV=2 (THE FIRST PRIME). (1) WE CHECK WHETHER DIV IS GREATER THAN THE LIMIT. IF IT IS SO,

//

AND AGAIN COMPARE THE NEXT PRIME WITH THE LIMIT AND SO ON. IF DIV DIVIDES THE NUMBER, WE STORE DIV AS THE FIRST PRIME FACTOR AND AGAIN DIVIDE THE QUOTIENT BY THE SAME DIV AS MANY TIMES AS WE FIND THAT DIV DIVIDES THE QUOTIENT. WE GO ON INCREASING THE POWER OF THE FIRST PRIME FACTOR. IF DIV DOES NOT DIVIDE THE QUOTIENT FURTHER, WE AGAIN FIND THE INTEGRAL PART OF THE SQUARE ROOT OF THE QUOTIENT. WE MAKE LIMIT EQUAL TO THIS NUMBER AND PUT DIV EQUAL TO THE NEXT PRIME. THEN WE REPEAT THE PROCESS FROM STEP (1). IF IN THE PROCESS OF DIVISION THE QUOTIENT BECOMES UNITY, WE TERMINATE THE PROCESS. THE PROCESS CAN ALSO BE TERMINATED IN ANOTHER WAY, IF IN THE END WE FIND THAT THE NEXT PRIME CALLED DIV IS GREATER THAN THE INTEGER PART OF THE SQUARE ROOT OF THE LAST QUOTIENT, IN WHICH CASE THE LAST QUOTIENT WILL BE A PRIME. ONCE ALL THE PRIME FACTORS TOGETHER WITH THEIR POWERS ARE OBTAINED, THE CALCULATION OF ALL FACTORS PRESENTS NO FURTHER DIFFICULTY.

THE EXECUTION OF THE SUBROUTINE CALLED FAC DEPENDS ON THE FACT THAT ALL PRIMES UPTO A CERTAIN LIMIT MUST BE STORED IN AN ARRAY IN ADVANCE. FOR THIS PURPOSE WE USE ANOTHER SUBROUTINE CALLED STORE, WHICH HAS BEEN WRITTEN BY MY TEACHER AND PUBLISHED IN THE PROCEEDINGS OF A WORKSHOP ON COMPUTER APPLICATIONS HELD IN ROORKEE UNIVERSITY IN 1986. WE REQUIRE ALL PRIMES UPTO THE SQUARE ROOT OF THE GREATEST INTEGER LIMIT OF THE COMPUTER ON WHICH WE ARE WORKING, PLUS ONE PRIME EXTRA. IN OUR CASE, THE MAXIMUM LIMIT IS 32767 AND THE INTEGER PART OF THE SQUARE ROOT IS 181. THE NEXT PRIME IS 191 AND SO WE HAVE TO STORE 43 PRIMES IN ALL FROM 2 THROUGH 191. THAT

//

IS WHY WE HAVE SPECIFIED THE SIZE OF THE ARRAY CONTAINING PRIMES AS 43 IN OUR PROGRAMS. IF WE WORK ON ANY OTHER COMPUTER, THE SIZE WILL CHANGE. ALSO FOR ANY NUMBER < 32768, THERE CAN BE MAXIMUM 6 DISTINCT PRIME FACTORS. SO WE HAVE SPECIFIED THE SIZES OF THE ARRAYS CONTAINING PRIME FACTORS OF A NUMBER AND CONTAINING POWERS OF THOSE FACTORS AS 6 EACH. FURTHER, THE MAXIMUM NUMBER OF ALL FACTORS FOR ANY NUMBER < 32768 IS 96. (ONE SUCH NUMBER IS 32760.) SO WE HAVE SPECIFIED THE SIZE AS 96 IN THE ARRAY WHICH CONTAINS ALL THE FACTORS OF ANY NATURAL NUMBER. OF COURSE, ALL THESE SIZES ARE SUBJECT TO CHANGE AND IF WE HAVE TO WORK THESE PROGRAMS ON ANY OTHER COMPUTER, THEY WILL BE CHANGED.

AFTER STORING ALL THE FACTORS OF THE LAST NON-ZERO COEFFICIENT OF THE POLYNOMIAL IN AN ARRAY, WE SORT THEM OUT IN ASCENDING ORDER BY USING 'SELECTION AND INTERCHANGE' METHOD IN ANOTHER SUBROUTINE CALLED SORT. THE COMPLETE PROGRAM FOR CALCULATING ALL INTEGER ZEROS OF A MONIC POLYNOMIAL, TOGETHER WITH ALL SUBPROGRAMS, NAMELY, FAC, STORE, SORT, SORUT AND POLDIV IS GIVEN ON THE SUCCEEDING PAGES. THE LAST SUBROUTINE CALLED POLDIV IS IN FACT A SUBROUTINE WHICH DIVIDES A MONIC POLYNOMIAL OF DEGREE 1 BY A LINEAR POLYNOMIAL OF THE FORM X-P.

```
      DIMENSION ICOF(25),ITZ(25),IFAC(96)
      INTEGER PR(43)
      COMMON/A1/PR/B1/ICOF,NC,IC/C1/IFAC,NF/D1/KK
      CALL STORE
      WRITE ( 1 , 7 )
7     FORMAT ( 1X , 'THIS PROGRAM FIND ALL INTEGER ZEROS OF A MONIC'
     1, 1X , 'POLYNOMIAL' )
19    WRITE(1,27)
27    FORMAT(1X,'TYPE THE DEGREE OF THE POLYNOMIAL IN FORMAT ( I2 )'
     1/ 1X , 'FOR STOPPING : PLEASE TYPE 0' )
      READ(1,47)ID
47    FORMAT(I2)
      IF(ID)89,799,140
89    WRITE(1,117)
117   FORMAT(1X,'DATA ILLEGAL, KINDLY TYPE  AGAIN')
      GO TO 19
C     TEST WHETHER THE GIVEN DEGREE WILL CROSS THE DIMENSION
140   IF(ID.GT.25)GO TO 669
      WRITE(1,167)
167   FORMAT(1X,'TYPE ALL THE COEFFICIENTS OF THE POLYNOMIAL' , 1X ,
     1'IN FORMAT ( 13I6 )' )
      READ(1,187)(ICOF(K),K=1,ID)
187   FORMAT( 13I6 )
      NC=ID
      NZ=0
C     TEST WHETHER 0 IS ONE OF THE ZEROS OF THE POLYNOMIAL
      DO 211 J=1,ID
      IF(ICOF(NC).NE.0)GO TO 242
      NZ=NZ+1
      ITZ(NZ)=0
211   NC=NC-1
      GO TO 450
C     IF THE DEGREE OF THE POLYNOMIAL IS 1, THEN THE NEGATIVE OF THE
C     CONSTANT TERM WILL BE THE LAST ZERO OF THE POLYNOMIAL
242   IF(NC.NE.1)GO TO 275
256   NZ=NZ+1
      ITZ(NZ)=-ICOF(1)
      GO TO 450
C     FIND ALL THE FACTORS OF THE LAST NON-ZERO COEFFICIENT
275   NEFC=0
      KK=IABS(ICOF(NC))
      CALL FAC
300   NEFC=NEFC+1
      IF(NEFC.GT.NF)GO TO 525
      IC=IFAC(NEFC)
C     CHECK WHETHER A PARTICULAR FACTOR SATISFIES THE POLYNOMIAL
335   ISUM=1
      DO 377 K=1,NC
377   ISUM=ISUM*IC+ICOF(K)
```

```
      IF(ISUM.EQ.0)GO TO 406
C     TEST WHETHER THIS FACTOR IS NEGATIVE
      IF(IC.LT.0)GO TO 300
      IC=-IC
      GO TO 335
406   NZ=NZ+1
      ITZ(NZ)=IC
      CALL POLDIV
      IF(NC.EQ.1)GO TO 256
      GO TO 335
450   WRITE(1,487)(ITZ(J),J=1,NZ)
487   FORMAT(1X,'ALL THE ZEROS OF THE POLYNOMIAL ARE INTEGERS AND ARE'
     1/(1X,11I7))
      GO TO 19
525   IF(NZ.EQ.0)GO TO 589
C     THE NUMBERS OF INTEGER ZEROS AND NON-INTEGER ZEROS ARE COMPUTED
      NUMZ=ID-NZ
      WRITE(1,567)NUMZ,NZ,(ITZ(J),J=1,NZ)
567   FORMAT(1X,'THERE ARE ',I2,' NON-INTEGER ZEROS OF THE POLYNOMIAL'
     1/1X,'THE ',I2,' INTEGER ZEROS OF THE POLYNOMIAL ARE'/
     2(1X,11I7))
      GO TO 19
589   WRITE(1,617)
617   FORMAT(1X,'THERE ARE NO INTEGER ZEROS OF THE POLYNOMIAL')
      GO TO 19
669   WRITE(1,697)
697   FORMAT(1X,'THE GIVEN DEGREE WILL CROSS THE DIMENSION,',
     1' CHANGE DIMENSION')
799   STOP
      END
C     THIS SUBROUTINE DIVIDES A MONIC POLYNOMIAL BY A LINEAR POLYNOMIAL
      SUBROUTINE POLDIV
      DIMENSION ICOF(25)
      COMMON/B1/ICOF,NC,ID
      NC=NC-1
      ICOF(1)=ICOF(1)+ID
      IF(NC.EQ.1)RETURN
      DO 122 I=2,NC
122   ICOF(I)=ICOF(I)+ID*ICOF(I-1)
      RETURN
      END
      SUBROUTINE FAC
      INTEGER PRIME(43),FACT(6),POWER(6),FACTOR(96),SQRUT,DIV,QUOT,
     1PROD,FACTM,POWERM
      COMMON/A1/PRIME/C1/FACTOR,N/D1/NUMBER
      IF(NUMBER.GT.1)GO TO 202
      N=1
      FACTOR(1)=1
      RETURN
202   N1=NUMBER
      J=1
```

```
      NOF=0
      DIV=2
      IND=1
245   I=1
C     FIND ALL PRIME FACTORS OF NUMBER WITH THEIR POWERS
      LIMIT=SQRUT(N1)
284   IF(DIV.GT.LIMIT)GO TO 475
295   QUOT=N1/DIV
      IF(N1.GT.QUOT*DIV)GO TO 450
      GO TO(306,418),I
306   NOF=NOF+1
      POWER(NOF)=1
      GO TO(352,553),IND
352   FACT(NOF)=DIV
      I=2
375   N1=QUOT
      IF(N1.EQ.1)GO TO 585
      GO TO 295
418   POWER(NOF)=POWER(NOF)+1
      GO TO 375
450   J=J+1
      DIV=PRIME(J)
      GO TO(284,245),I
475   IND=2
      GO TO 306
553   FACT(NOF)=N1
585   M=1
      N=1
      FACTOR(1)=1
      PROD=1
C     FIND ALL THE FACTORS OF NUMBER
620   FACTM=FACT(M)
      POWERM=POWER(M)
      LIMIT=PROD*POWERM
      DO 677 J1=1,LIMIT
      N=N+1
677   FACTOR(N)=FACTOR(N-PROD)*FACTM
      IF(M.EQ.NOF)GO TO 700
      PROD=PROD+LIMIT
      M=M+1
      GO TO 620
700   IF(NOF.GT.1)CALL SORT
      RETURN
      END
C     THIS SUBROUTINE GENERATES ALL THE PRIME NUMBERS UPTO A CERTAIN
C     LIMIT
      SUBROUTINE STORE
      INTEGER P(43),PRMSQ,PRMPRD,DIV,PMP1
      COMMON/A1/P
      P(1)=2
```

```
      P(2)=3
      P(3)=5
      P(4)=7
      P(5)=11
      P(6)=13
      K=6
      M=2
      PRMSQ=25
      PRMPRD=35
      IND=1
      N=17
202   GO TO(240,270),IND
240   IF(N.LT.PRMSQ)GO TO 310
      IND=2
      GO TO 500
270   IF(N.EQ.PRMPRD)GO TO 480
310   DO 344 J=2,M
      DIV=P(J)
      IF(N.EQ.N/DIV*DIV)GO TO 500
344   CONTINUE
      K=K+1
      P(K)=N
      IF(N.GT.181)RETURN
C     MAXIMUM LIMIT FOR THIS PARTICULAR COMPUTER IS 181
      GO TO 500
480   IND=1
      M=M+1
      PMP1=P(M+1)
      PRMSQ=PMP1*PMP1
      PRMPRD=PMP1*P(M+2)
500   N=N+2
      GO TO 202
      END
C     THIS PROGRAM CALCULATES THE INTEGER PART OF THE SQUARE ROOT OF A
C     NUMBER BY ARITHMETICAL METHOD
      INTEGER FUNCTION SQRUT(N)
      INTEGER PAIR(3),QUOT,PAIRI,SQRUT2,REM,DVND,TRDIV,TRQT
      NN=N
      I=1
C     MAKE PAIRS STARTING FROM THE RIGHT
202   QUOT=NN/100
      PAIR(I)=NN-100*QUOT
      IF(QUOT.EQ.0)GO TO 226
      I=I+1
      NN=QUOT
      GO TO 202
C     CALCULATE THE INTEGER PART OF THE SQUARE ROOT
226   PAIRI=PAIR(I)
      DO 233 NLS=1,8
      SQRUT=10-NLS
```

```
      SQRUT2=SQRUT*SQRUT
      IF(PAIRI.GE.SQRUT2)GO TO 253
  233 CONTINUE
      SQRUT=1
  253 REM=PAIRI-SQRUT*SQRUT
  353 IF(I.EQ.1)RETURN
      I=I-1
      DVND=REM*100+PAIR(I)
      TRDIV=20*SQRUT
      TROT=DVND/TRDIV
      IF(TROT.GT.9)TROT=9
  361 REM=DVND-(TRDIV+TROT)*TROT
      IF(REM.GE.0)GO TO 400
      TROT=TROT-1
      GO TO 361
  400 SQRUT=SQRUT*10+TROT
      GO TO 353
      END
C     THIS SUBROUTINE ARRANGES A GIVEN SET OF NUMBERS IN ASCENDING
C     ORDER
      SUBROUTINE SORT
      INTEGER FAC(96),SMALL,COPY
      COMMON/C1/FAC,L
      LM1=L-1
      DO 244 I=1,LM1
      IP1=I+1
      NSMALL=IP1
      SMALL=FAC(NSMALL)
      IF(I.EQ.LM1)GO TO 183
      IP2=I+2
      DO 122 K=IP2,L
      IF(SMALL.LE.FAC(K))GO TO 122
      NSMALL=K
      SMALL=FAC(NSMALL)
  122 CONTINUE
  183 IF(SMALL.GE.FAC(I))GO TO 244
      COPY=FAC(I)
      FAC(I)=SMALL
      FAC(NSMALL)=COPY
  244 CONTINUE
      RETURN
      END

//
```

ALGORITHM FOR CALCULATING ALL RATIONAL ZEROS OF A POLYNOMIAL (NOT NECESSARILY MONIC) WITH INTEGER COEFFICIENTS

Assume that the zeros of the polynomial are $x_1, x_2, x_3, \ldots, x_n$. If we multiply them by a natural number k and form another polynomial whose zeros are $kx_1, kx_2, kx_3, \ldots, kx_n$, then by a suitable choice of k, the other polynomial can be monic, still having all integer co-efficients. In fact, if the original polynomial is

$$a_o x^n + a_1 x^{n-1} + a_2 x^{n-2} + \ldots + a_{n-1} x + a_n,$$

where $n \in N$ and $a_o, a_1, a_2, \ldots, a_{n-1}, a_n \in Z$, then it is easy to see that if we multiply all its zeros by $|a_o|$, then the other polynomial having its zeros as $|a_o| x_1, |a_o| x_2, |a_o| x_3, \ldots |a_o| x_n$ will certainly be monic and have all its coefficients in Z. Notwithstanding this fact, in most cases we can find a natural number k, much smaller than a_o, which will do the job equally well. For example, we consider the polynomial

$$-24x^3 + 84x^2 - 126x - 27.$$

In this case $|a_o| = |-24| = 24$. But instead of multiplying all its zeros by 24, if we multiply them by 2, we get the polynomial

$$-24x^3 + 168x^2 - 504x - 216$$

or, equivalently,

$$x^3 - 7x^2 + 21x + 9,$$

which is monic and has all its coefficients in Z. So first of all we develop an algorithm for calculating the least natural number which will do the trick. In fact, if all zeros of a particular polynomial happen to be rational and if we express them in lowest terms, this number k will be just the L.C.M. of the denominators of all the zeros. But in the general case we cannot make this statement. Still the existence of k is beyond doubt. At worst, k can be $|a_o|$.

Let the given polynomial be

$$a_o x^n + a_1 x^{n-1} + a_2 x^{n-2} + \ldots + a_{n-1} x + a_n, \qquad (1)$$

where we assume that $n \in N$, $a_o > o$, $a_o, a_1, a_2, \ldots, a_n \in Z$, and $(a_o, a_1, a_2, \ldots, a_n) = 1$. If $a_o = 1$, k will be 1 and we are through.

Otherwise let $a_o = \prod_{i=1}^{r} p_i^{e_i}$, where $r \in N$, p_i is a prime for all $i \in \{1,2,3,\ldots, r\}$ and $e_i \in N$ for all $i \in \{1,2,3,\ldots, r\}$.

Naturally k will also be of the form

$k = \prod_{i=1}^{r} p_i^{f_i}$, where $1 \leq f_i \leq e_i \ \forall\ i \in \{1,2,3, \ldots, r\}$. To fix the ideas, we assume that $r = 1$, so that $a_o = p^e$, $k = p^f$, $1 \leq f \leq e$, p being a prime. Then, after putting $kx = y$, the given polynomial becomes

$$a_o \left(\frac{y}{k}\right)^n + a_1 \left(\frac{y}{k}\right)^{n-1} + \ldots + a_{n-1}\left(\frac{y}{k}\right) + a_n ,$$

which has the same zeros as the polynomial

$$y^n + \frac{a_1 k}{a_o} y^{n-1} + \frac{a_2 k^2}{a_o} y^{n-2} + \ldots + \frac{a_{n-1} k^{n-1}}{a_o} y + \frac{a_n k^n}{a_o} \quad \ldots\ldots\ldots (2)$$

Now assume that $p^{m_1} \,||\, a_1$, $p^{m_2} \,||\, a_2, \ldots p^{m_n} \,||\, a_n$,

where $m_i > o \quad i \in \{1,2,3,\ldots,n\}$

By $p^t \,||\, N$ we mean that $p^t \mid N$ but $p^{t+1} \nmid N$. For example, we say that $2^5 \,||\, 96$ because $2^5 \mid 96$ but $2^6 \nmid 96$. If all coefficients of the polynomial (2) are in $\mathbb{Z}$, then we must have

$$m_1 + f \geq e, \quad m_2 + 2f \geq e, \quad m_3 + 3f \geq e, \ldots, \quad m_n + nf \geq e$$

$$\Rightarrow \quad e-f \leq m_1, \quad e-2f \leq m_2, \quad e-3f \leq m_3, \ldots, \quad e-nf \leq m_n .$$

$$\Rightarrow \quad f \geq e-m_1, \quad 2f \geq e-m_2, \quad 3f \geq e-m_3, \ldots, \quad nf \geq e-m_n$$

$$\Rightarrow \quad f \geq e-m_1, \quad f \geq \frac{e-m_2}{2}, \quad f \geq \frac{e-m_3}{3}, \ldots f \geq \frac{e-m_n}{n},$$

$$\Rightarrow f \geq -\left[\frac{m_i - e}{i}\right] \quad \forall\, i \in \{1,2,3,\ldots, n\}$$

$$\Rightarrow f = \max_{1 \leq i \leq n} \left\{-\left[\frac{m_i - e}{i}\right]\right\} .$$

If any $m_i \geq e$, we can omit it while taking the maximum, because $f > 0$ is evident. Similarly, if any m_i is such that $e > m_i \geq e-i$, we can omit it also, because in that case we shall get $-\left[\frac{m_i - e}{i}\right] = 1$ and $f \geq 1$ is also otherwise evident. So we should consider only those m_i, if any, which are less than $e-i$. If there is no such m_i, we must

take $f = 1$, $k = p$. Otherwise we should write

$$f = \max_{\substack{i \leq i \leq n \\ m_i \leq e-i}} -\left[\frac{m_i - e}{i}\right], \quad k = p^f$$

(We have assumed that the H.C.F. of $a_o, a_1, a_2, \ldots, a_n$ is unity. So at least 1 m_i should be zero.)

We can repeat this process for every prime p_i and obtain $f_i \varepsilon N$ for every value of i from 1 to r, both inclusive. Then we can write

$$k = \prod_{i=1}^{r} p_i^{f_i} .$$

These computations can be easily carried out by a nest of 2 DO loops, 1 for r and 1 for n. While we are in the DO loop, we shall ignore all coefficients, if any, which happen to be zero.

The program for finding k is given on the succeeding pages.

```
      DIMENSION ICOF ( 26 ) , INCO ( 26 ) , IFAC ( 6 ) , NF ( 6 )
      INTEGER PR ( 43 )
      COMMON / A1 / PR / C1 / IFAC , NF , NT / D1 / KK
      CALL STORE
      WRITE ( 1 , 7 )
 7    FORMAT ( 1X , 'THIS PROGRAM FINDS THE LEAST NATURAL NUMBER'
     1 , 1X , 'BY WHICH ALL ZEROS' / 1X, 'OF A POLYNOMIAL SHOULD BE'
     1, 1X , 'MULTIPLIED SO THAT THE POLYNOMIAL RECONSTRUCTED FROM'
     1/ 1X , 'THE NEW ZEROS MIGHT BE MONIC' )
 79   WRITE ( 1 , 107 )
 107  FORMAT ( 1X , 'TYPE THE DEGREE OF THE POLYNOMIAL IN FORMAT (I2)')
 791  READ ( 1 , 127 ) ID
 127  FORMAT ( I2 )
      IF ( ID ) 149 , 599 , 205
 149  WRITE ( 1 , 167 )
 167  FORMAT ( 1X , 'DATA ILLEGAL , KINDLY TYPE AGAIN' )
      GO TO 791
 205  IF ( ID .GT. 25 ) GO TO 549
      NC = ID + 1
      WRITE ( 1 , 287 )
 287  FORMAT ( 1X , 'TYPE ALL COEFFICIENTS OF THE POLYNOMIAL IN'
     1, 1X , 'FORMAT ( 13I6 )' )
 7000 READ ( 1 , 327 ) ( ICOF ( J ) , J = 1 , NC )
 327  FORMAT ( 13I6 )
      IF ( ICOF ( 1 ) .EQ. 0 ) GO TO 1491
      WRITE ( 1 , 768 )
 768  FORMAT ( 1X , 'THE COEFFICIENTS OF THE POLYNOMIAL ARE GIVEN' ,
     11X , 'BELOW:-' )
      WRITE ( 1 , 337 ) ( ICOF ( J ) , J = 1 , NC )
 337  FORMAT ( 1X , 11I7 )
      LN = 1
C     IF THE LEADING COEFFICIENT IS 1 OR -1, THEN LN = 1
      IF ( IABS ( ICOF ( 1 ) ) .EQ. 1 ) GO TO 489
      NHCF = NHCF1 ( ICOF , NC )
      IF ( NHCF .EQ. IABS ( ICOF ( 1 ) ) ) GO TO 489
      DO 355 J = 1 , NC
 355  INCO ( J ) = ICOF ( J ) / NHCF
      KK = IABS ( INCO ( 1 ) )
      CALL FAC
C     FOR A PARTICULAR PRIME FACTOR, CALCULATE THE LEAST POWER
      DO 488 K = 1 , NT
      NF = 1
      IP = IFAC ( K )
      IE = NF ( K )
      DO 455 I = 2 , NC
      ICAR = I - 1
      IF ( ICAR .EQ. IE ) GO TO 488
      IF ( INCO ( I ) .EQ. 0 ) GO TO 455
      M = 0
```

```
      ITEST = INCO ( I )
370   ICHEK = ITEST / IP
      IF ( ITEST .NE. IP * ICHEK ) GO TO 418
      ITEST = ICHEK
      M = M + 1
C     CHECK WHETHER THE COEFFICIENT PLUS THE NUMBER OF TIMES IT IS
C     ALREADY DIVISIBLE IS EQUAL TO THE POWER OF THE FACTOR
      IF ( M + ICAR .EQ. IE ) GO TO 455
      GO TO 370
C     CALCULATE THE HIGHEST POWER OF THE CURRENT PRIME FACTOR
418   L = ( IE - M ) / ( I - 1 )
      IF ( L * ( I - 1 ) .NE. IE - M ) L = L + 1
      IF ( L .GT. KF ) KF = L
455   CONTINUE
488   LN = LN * IP ** KF
489   WRITE ( 1 , 517 ) LN
517   FORMAT ( 1X , 'THE LEAST NUMBER REQUIRED =' , 1X , I5 )
      GO TO 79
549   WRITE ( 1 , 567 )
567   FORMAT ( 1X , 'THE GIVEN NUMBER WILL CROSS DIMENSION,' , 1X ,
     1'CHANGE DIMENSION' )
599   STOP
1491  WRITE ( 1 , 800 )
800   FORMAT ( 1X , 'THE FIRST COEFFICIENT OF A POLYNOMIAL CANNOT BE'
     1, 1X , 'ZERO' / 1X , 'KINDLY TYPE ALL COEFFICIENTS AGAIN' )
      GO TO 7000
      END
      FUNCTION NHCF1 ( ICOF , NC )
      INTEGER HCF
      DIMENSION ICOF ( 26 )
      NHCF1 = IABS ( ICOF ( 1 ) )
      DO 122 J = 2 , NC
122   NHCF1 = HCF ( NHCF1 , ICOF ( J ) )
      RETURN
      END
      INTEGER FUNCTION HCF ( I , J )
      INTEGER REM
      IF ( I .NE. 0 ) GO TO 102
      HCF = IABS ( J )
      RETURN
102   IF ( J .NE. 0 ) GO TO 153
      HCF = IABS ( I )
      RETURN
153   KOPY1 = IABS ( I )
      KOPY2 = IABS ( J )
180   REM = MOD ( KOPY1 , KOPY2 )
      IF ( REM .NE. 0 ) GO TO 200
      HCF = KOPY2
      RETURN
200   KOPY1 = KOPY2
      KOPY2 = REM
```

```
      GO TO 180
      END
C     SUBROUTINE FAC FINDS THE PRIME FACTORS OF A NATURAL NUMBER
C     TOGETHER WITH THEIR PRIMES
      SUBROUTINE FAC
      INTEGER PRIME(43),FACT(6),POWER(6),SQRUT,DIV,QUOT
      COMMON/ A1 / PRIME / C1 / FACT, POWER, NOF / D1 / NUMBER
      N1=NUMBER
      J=1
      NOF=0
      DIV=2
      IND=1
245   I=1
      LIMIT=SQRUT(N1)
284   IF(DIV.GT.LIMIT)GO TO 475
295   QUOT=N1/DIV
      IF(N1.GT.QUOT*DIV)GO TO 450
      GO TO(306,418),I
306   NOF=NOF+1
      POWER(NOF)=1
      GO TO(352,553),IND
352   FACT(NOF)=DIV
      I=2
375   N1=QUOT
      IF(N1.EQ.1) RETURN
      GO TO 295
418   POWER(NOF)=POWER(NOF)+1
      GO TO 375
450   J=J+1
      DIV=PRIME(J)
      GO TO(284,245),I
475   IND=2
      GO TO 306
553   FACT(NOF)=N1
      RETURN
      END
      SUBROUTINE STORE
      INTEGER P(43),PRMSQ,PRMPRD,DIV,PMP1
      COMMON/A1/P
      P(1)=2
      P(2)=3
      P(3)=5
      P(4)=7
      P(5)=11
      P(6)=13
      K=6
      M=2
      PRMSQ=25
      PRMPRD=35
      IND=1
      N=17
```

```
202     GO TO(240,270),IND
240     IF(N.LT.PRMSQ)GO TO 310
        IND=2
        GO TO 500
270     IF(N.EQ.PRMPRD)GO TO 480
310     DO 344 J=2,M
        DIV=P(J)
        IF(N.EQ.N/DI   DIV)GO TO 500
344     CONTINUE
        K=K+1
        P(K)=N
        IF(N.GT.181)RETURN
        GO TO 500
480     IND=1
        M=M+1
        PMP1=P(M+1)
        PRMSQ=PMP1*PMP1
        PRMPRD=PMP1*P(M+2)
500     N=N+2
        GO TO 202
        END
        INTEGER FUNCTION SQRUT(N)
        INTEGER PAIR(3),QUOT,PAIRI,SQRUT2,REM,DVND,TRDIV,TRQT
        NN=N
        I=1
202     QUOT=NN/100
        PAIR(I)=NN-100*QUOT
        IF(QUOT.EQ.0)GO TO 226
        I=I+1
        NN=QUOT
        GO TO 202
226     PAIRI=PAIR(I)
        DO 233 NLS=1,8
        SQRUT=10-NLS
        SQRUT2=SQRUT*SQRUT
        IF(PAIRI.GE.SQRUT2)GO TO 253
233     CONTINUE
        SQRUT=1
253     REM=PAIRI-SQRUT*SQRUT
353     IF(I.EQ.1)RETURN
        I=I-1
        DVND=REM*100+PAIR(I)
        TRDIV=20*SQRUT
        TRQT=DVND/TRDIV
        IF(TRQT.GT.9)TRQT=9
361     REM=DVND-(TRDIV+TRQT)*TRQT
        IF(REM.GE.0)GO TO 400
        TRQT=TRQT-1
        GO TO 361
400     SQRUT=SQRUT*10+TRQT
        GO TO 353
        END
```

Once we have calculated k, we calculate the coefficients of the new polynomial whose zeros are k times the zeros of the given polynomial. We divide all coefficients by the leading coefficients. All the new coefficients will be integers even after division. Now we can use the previous program for calculating all its integer zeros, because the new polynomial is monic. Then we divide all zeros by k and get the rational zeros of the original polynomial. For example if the integer zeros are - 8, 9, 15, and k = 6, the zeros of the original polynomial will be $-\frac{4}{3}$, $\frac{3}{2}$ and $\frac{5}{2}$.

The program for calculating the rational zeros is given on the succeeding pages. Here we have made use of the program for calculating k (which we have changed into a FUNCTION Subprogram), and also of the earlier program for calculating all integer zeros of a monic polynomial (which we have changed into a SUBROUTINE Subprogram). In the end we divide all the zeros by k through a DO loop and before printing them, we reduce them back to lowest terms, using the FUNCTION Subprogram called HCF.

```
      INTEGER HCF
      DIMENSION INCOF ( 26 ) , NUM ( 25 ) , IDEN ( 25 ) , IP ( 43 )
      DIMENSION NEWCOF ( 25 )
      COMMON / A1 / IP
      WRITE ( 1 , 100 )
100   FORMAT ( 1X , 'THIS PROGRAM CALCULATES ALL RATIONAL ZEROS OF A
     11X , 'POLYNOMIAL' / 1X , 'WITH INTEGER COEFFICIENTS ' )
      CALL STORE
6     WRITE ( 1 , 200 )
200   FORMAT ( 1X , 'TYPE THE DEGREE OF THE POLYNOMIAL IN FORMAT(I2)'
69    READ ( 1 , 3 ) ID
3     FORMAT ( I2 )
      IF ( ID ) 61 , 62 , 63
61    WRITE ( 1 , 500 )
500   FORMAT ( 1X , 'DATA ILLEGAL, PLEASE TYPE AGAIN' )
      GO TO 69
63    IF ( ID .GT. 25 ) GO TO 75
      NC = ID + 1
      WRITE ( 1 , 900 )
900   FORMAT ( 1X , 'TYPE ALL COEFFICIENTS OF THE POLYNOMIAL IN' ,
     11X , 'FORMAT ( 13I6 )' )
2900  READ ( 1 , 91 ) ( INCOF ( I ) , I = 1 , NC )
91    FORMAT ( 13I6 )
      IF ( INCOF ( 1 ) .NE. 0 ) GO TO 205
      WRITE ( 1 , 275 )
275   FORMAT ( 1X , 'THE LEADING COEFFICIENT CANNOT BE ZERO, PLEASE'
     11X ,' 'TYPE ALL COEFFICIENTS AGAIN' )
      GO TO 2900
C     CHANGE THE LEAST VALUE PROGRAM INTO A FUNCTION SUBPROGRAM ANI
C     CALL IT KP
205   WRITE ( 1 , 888 )
888   FORMAT ( 1X , 'THE COEFFICIENTS OF THE POLYNOMIAL ARE GIVEN' ,
     11X , 'BELOW:-' )
      WRITE ( 1 , 889 ) ( INCOF ( I ) , I = 1 , NC )
889   FORMAT ( 1X , 11I7 )
      N = KP ( INCOF , NC )
      DO 5081 I = 2 , NC
5081  INCOF ( I ) = INCOF ( I ) * N ** ( I - 1 )
      KDIV = INCOF ( 1 )
      DO 5082 I = 1 , ID
5082  NEWCOF ( I ) = INCOF ( I + 1 ) / KDIV
C     CHANGE THE PROGRAM FOR CALCULATING ALL ZEROS OF A MONIC
C     POLYNOMIAL INTO A SUBROUTINE SUBPROGRAM AND NAME IT SUBR
      CALL SUBR ( NEWCOF , ID , NUM , M )
      IF ( M .EQ. 0 ) GO TO 15
      IF ( M .LT. ID ) GO TO 25
      WRITE ( 1 , 71 )
71    FORMAT ( 1X , 'ALL ZEROS OF THE POLYNOMIAL ARE RATIONAL'
251   DO 513 KK = 1 , ID
```

```
      NHCF = HCF ( NUM ( KK ) , N )
      NUM   FF ) = NUM ( FF ) / NHCF
  510 IDEN   FF ) = N   NHCF
      WRITE ( 1 , 512
  512 FORMAT ( 1X , 'THE NUMERATORS OF THE ZEROS ARE GIVEN BELOW:  '
      WRITE ( 1 , 885 )  NUM ( I ) , I = 1 , ID )
      WRITE   1 , 513
  513 FORMAT ( 1X , 'THE DENOMINATORS OF THE ZEROS ARE GIVEN BELOW:-')
      WRITE ( 1 , 885 ) ( IDEN ( I ) , I = 1 , ID )
      GO TO 6
   25 WRITE ( 1 , 7002 ) M
 7002 FORMAT ( 1X , 'ONLY ' , I2 , 1X , 'ZEROS OF THE POLYNOMIAL',
     1 1X , 'ARE RATIONAL '
      ID = M
      GO TO 251
   15 WRITE ( 1 , 7050 )
 7050 FORMAT ( 1X , 'THE POLYNOMIAL HAS NO RATIONAL ZEROS' )
      GO TO 6
   75 WRITE ( 1 , 756 )
  756 FORMAT (  THIS DEGREE WILL CROSS THE DIMENSION,' ,
     1 1X , 'CHANGE DIMENSION  )
   62 STOP
      END
      FUNCTION FF ( INCOF , NC )
      DIMENSION INCOF ( 26 ) , IFAC  ( ) , NF ( 6 )
      COMMON  B1  IFAC , NF , NT  D1  FF / F1 / INDEX
      FF = 1
      IF ( IABS ( INCOF ( 1 ) ) .EQ. 1 ) RETURN
      NHCF = NHCF1 ( INCOF , NC )
      IF ( NHCF .EQ. IABS ( INCOF ( 1 ) ) ) RETURN
      DO 355 J = 1 , NC
  355 INCOF ( J ) = INCOF ( J ) / NHCF
      FF = IABS ( INCOF ( 1 ) )
      INDEX = 1
      CALL FAC
      DO 488 K = 1 , NT
      FF = 1
      IP = IFAC ( K )
      IE = NF ( K )
      DO 455 I = 2 , NC
      ICAR = I - 1
      IF ( ICAR .EQ. IE ) GO TO 488
      IF ( INCOF ( I ) .EQ. 0 ) GO TO 455
      M = 0
      ITEST = INCOF ( I )
  370 ICHEK = ITEST / IP
      IF ( ITEST .NE. IP * ICHEK ) GO TO 418
      ITEST = ICHEK
      M = M + 1
      IF ( M + ICAR .EQ. IE ) GO TO 455
```

```
      GO TO 370
418   L = ( IE - M ) / ( I - 1 )
      IF ( L * ( I - 1 ) .NE. IE - M ) L = L + .
      IF ( L .GT. IF ) IF = L
455   CONTINUE
482   IF = IF * IP ** IF
      RETURN
      END
      FUNCTION NHCF1 ( ICOF , NC )
      INTEGER HCF
      DIMENSION ICOF ( 26 )
      NHCF1 = IABS ( ICOF ( 1 ) )
      DO 122 J = 2 , NC
122   NHCF1 = HCF ( NHCF1 , ICOF ( J ) )
      RETURN
      END
      INTEGER FUNCTION HCF ( I , J )
      INTEGER REM
      IF ( I .NE. 0 ) GO TO 102
      HCF = IABS ( J )
      RETURN
102   IF ( J .NE. 0 ) GO TO 153
      HCF = IABS ( I )
      RETURN
153   KOPY2 = IABS ( I
      KOPY1 = IABS ( J )
180   REM = MOD ( KOPY2 , KOPY1 )
      IF ( REM .NE. 0 ) GO TO 200
      HCF = KOPY1
      RETURN
200   KOPY2 = KOPY1
      KOPY1 = REM
      GO TO 180
      END
      SUBROUTINE SUBR ( ICOF , ID , ITZ , NZ )
      DIMENSION ICOF ( 25 ) , ITZ ( 25 ) , IFAC ( 96 )
      COMMON / C1 / IFAC , NF / D1 / KK / B1 / NC , IC / F1 / INDEX
      NC = ID
      NZ = 0
      DO 211 J = 1 , ID
      IF ( ICOF ( NC ) .NE. 0 ) GO TO 242
      NZ = NZ + 1
      ITZ ( NZ ) = 0
211   NC = NC - 1
      RETURN
242   IF ( NC .NE. 1 ) GO TO 275
256   NZ = NZ + 1
      ITZ ( NZ ) = - ICOF ( 1 )
      RETURN
275   NEFC = 0
      KK = IABS ( ICOF ( NC ) )
```

```
      INDEX = 0
      CALL FAC
300   NEFC = NEFC + 1
      IF ( NEFC .GT. NF ) RETURN
      IC = IFAC ( NEFC )
335   ISUM = 1
      DO 377 I = 1 , NC
377   ISUM = ISUM * IC + ICOF ( I )
      IF ( ISUM .EQ. 0 ) GO TO 406
      IF ( IC .LT. 0 ) GO TO 300
      IC = - IC
      GO TO 335
406   NZ = NZ + 1
      ITZ ( NZ ) = IC
      CALL POLDIV ( ICOF )
      IF ( NC .EQ. 1 ) GO TO 256
      GO TO 335
      END
      SUBROUTINE POLDIV ( ICOF )
      DIMENSION ICOF ( 25 )
      COMMON / B1 / NC , ID
      NC = NC - 1
      ICOF ( 1 ) = ICOF ( 1 ) + ID
      IF ( NC .EQ. 1 ) RETURN
      DO 122 I = 2 , NC
122   ICOF ( I ) = ICOF ( I ) + ID * ICOF ( I - 1 )
      RETURN
      END
      SUBROUTINE FAC
      INTEGER PRIME ( 43 ) , FACT ( 6 ) , POWER ( 6 ) , FACTOR ( 96 ) ,
     1SQRUT , DIV , QUOT , PROD , FACTM , POWERM
      COMMON / A1 / PRIME / C1 / FACTOR , N / D1 / NUMBER / E1 /
     1FACT , POWER , NOF / F1 / INDEX
      IF ( NUMBER .GT. 1 ) GO TO 202
      N = 1
      FACTOR ( 1 ) = 1
      RETURN
202   N1 = NUMBER
      J = 1
      NOF = 0
      DIV = 2
      IND = 1
245   I = 1
      LIMIT = SQRUT ( N1 )
284   IF ( DIV .GT. LIMIT ) GO TO 475
295   QUOT = N1 / DIV
      IF ( N1 .GT. QUOT * DIV ) GO TO 450
      GO TO ( 306 , 418 ) , I
306   NOF = NOF + 1
      POWER ( NOF ) = 1
```

```
      GO TO ( 352 , 553 ) , IND
352   FACT ( NOF ) = DIV
      I = 2
375   N1 = QUOT
      IF ( N1 .EQ. 1 ) GO TO 585
      GO TO 295
416   POWER ( NOF ) = POWER ( NOF ) + 1
      GO TO 375
450   J = J + 1
      DIV = PRIME ( J )
      GO TO ( 284 , 245 ) , I
475   IND = 2
      GO TO 306
553   FACT ( NOF ) = N1
585   IF ( INDEX .EQ. 1 ) RETURN
      M = 1
      N = 1
      FACTOR ( 1 ) = 1
      PROD = 1
620   FACTM = FACT ( M )
      POWERM = POWER ( M )
      LIMIT = PROD * POWERM
      DO 677 J1 = 1 , LIMIT
      N = N + 1
677   FACTOR ( N ) = FACTOR ( N - PROD ) * FACTM
      IF ( M .EQ. NOF ) GO TO 700
      PROD = PROD + LIMIT
      M = M + 1
      GO TO 620
700   IF ( NOF .GT. 1 ) CALL SORT
      RETURN
      END
      SUBROUTINE STORE
      INTEGER P ( 43 ) , PRMSQ , PRMPRD , DIV , PMP1
      COMMON / A1 / P
      P ( 1 ) = 2
      P ( 2 ) = 3
      P ( 3 ) = 5
      P ( 4 ) = 7
      P ( 5 ) = 11
      P ( 6 ) = 13
      K = 6
      M = 2
      PRMSQ = 25
      PRMPRD = 35
      IND = 1
      N = 17
202   GO TO ( 240 , 270 ), IND
240   IF ( N .LT. PRMSQ ) GO TO 310
      IND = 2
      GO TO 500
```

```
270     IF ( N .EQ. PRMPRD ) GO TO 480
310     DO 344 J = 2 , M
        DIV = P ( J )
        IF ( N .EQ. N / DIV * DIV ) GO TO 500
344     CONTINUE
        I = I + 1
        P ( I ) = N
        IF ( N .GT. 181 ) RETURN
        GO TO 500
480     IND = 1
        M = M + 1
        PMP1 = P ( M + 1 )
        PRMSQ = PMP1 * PMP1
        PRMPRD = PMP1 * P ( M + 2 )
500     N = N + 2
        GO TO 202
        END
        INTEGER FUNCTION SQRUT ( N )
        INTEGER PAIR ( 3 ) , QUOT , PAIRI , SQRUT2 , REM , DVND , TRDIV ,
       1TRQT
        NN = N
        I = 1
202     QUOT = NN / 100
        PAIR ( I ) = NN - 100 * QUOT
        IF ( QUOT .EQ. 0 ) GO TO 226
        I = I + 1
        NN = QUOT
        GO TO 202
226     PAIRI = PAIR ( I )
        DO 233 NLS = 1 , 8
        SQRUT = 10 - NLS
        SQRUT2 = SQRUT * SQRUT
        IF ( PAIRI .GE. SQRUT2 ) GO TO 253
233     CONTINUE
        SQRUT = 1
253     REM = PAIRI - SQRUT * SQRUT
353     IF ( I .EQ. 1 ) RETURN
        I = I - 1
        DVND = REM * 100 + PAIR ( I )
        TRDIV = 20 * SQRUT
        TRQT = DVND / TRDIV
        IF ( TRQT .GT. 9 ) TRQT = 9
361     REM = DVND - ( TRDIV + TRQT ) * TRQT
        IF ( REM .GE. 0 ) GO TO 400
        TRQT = TRQT - 1
        GO TO 361
400     SQRUT = SQRUT * 10 + TRQT
        GO TO 353
        END
        SUBROUTINE SORT
```

```
      INTEGER FAC ( 96 ) , SMALL , COPY
      COMMON / C1 / FAC , L
      LM1 = L - 1
      DO 244 I = 1 , LM1
      IP1 = I + 1
      NSMALL = IP1
      SMALL = FAC ( NSMALL )
      IF ( I .EQ. LM1 ) GO TO 183
      IP2 = I + 2
      DO 122 K = IP2 , L
      IF ( SMALL .LE. FAC ( K ) ) GO TO 122
      NSMALL = K
      SMALL = FAC ( NSMALL )
122   CONTINUE
183   IF ( SMALL .GE. FAC ( I ) ) GO TO 244
      COPY = FAC ( I )
      FAC ( I ) = SMALL
      FAC ( NSMALL ) = COPY
244   CONTINUE
      RETURN
      END

//
```

CHAPTER 3

PROGRAMS GENERALISING THE DO LOOP STRUCTURE

1. INTRODUCTION

IN THIS CHAPTER WE SHALL DEVELOP A FEW PROGRAMS WHICH ATTEMPT TO GENERALISE THE STRUCTURE OF NESTED DO LOOPS TO SOME EXTENT. AS WE ARE AWARE, THE DO STATEMENT IS THE MOST POWERFUL CONTROL STATEMENT OF THE FORTRAN LANGUAGE. BUT WHILE FORMING DO LOOPS IN OUR PROGRAMS, WE HAVE TO FOLLOW MANY CONSTRAINTS. IN SOME CASES SUCH CONSTRAINTS BECOME SO SEVERE THAT OUR OBJECTIVES ARE ONLY PARTIALLY FULFILLED. FOR EXAMPLE, WE CAN, AT LEAST THEORETICALLY, PUT A NEST OF AS MANY DO LOOPS IN OUR PROGRAMS AS WE LIKE. SUCH OCCASIONS ARISE WHEN WE HAVE TO COMPUTE A FUNCTION OR MANY FUNCTIONS OF A LARGE NUMBER OF VARIABLES, WHEN EACH VARIABLE TAKES UP MANY VALUES AT EQUALLY SPACED INTERVALS, AND ALL VARIABLES CAN VARY INDEPENDENTLY OF EACH OTHER. FOR EXAMPLE, WE MAY LIKE TO EVALUATE SOME FUNCTION(S) OF X,Y,Z, WHERE X=1.2(0.1)2.5, Y=5(3)17, Z=-4.5(1)3.5. IN THIS CASE WE SHALL CONSTRUCT A NEST OF 3 DO LOOPS. IN FORTRAN IV THERE IS A RESTRICTION THAT ALL THE INDICES OF THE DO LOOPS MUST BE STRICTLY POSITIVE AND MOREOVER THEY MUST BE INTEGERS. ALSO ALL THE INDEXING PARAMETERS MUST BE STRICTLY POSITIVE. FURTHER, THE CALCULATION MUST PROCEED IN INCREASING ORDER OF EACH INDEX. FOR EXAMPLE, WE ARE FORBIDDEN TO WRITE A STATEMENT OF THIS KIND:-

```
DO 156 I = 15 , 8 , -2
```

WE MUST AVOID LOOPS IN WHICH A PARTICULAR INDEX MOVES BACKWARDS.

SOME OF THESE RESTRICTIONS HAVE, HOWEVER, BEEN REMOVED IN THE LATEST VERSION OF FORTRAN, NAMELY FORTRAN 77. IN THIS VERSION WE ALLOW OUR INDICES TO TAKE REAL AND DOUBLE PRECISION VALUES ALSO. MOREOVER, ANY OR ALL OF THE INDEXING PARAMETERS CAN BE ZERO OR NEGATIVE IN FORTRAN 77. THEREFORE OUR CALCULATIONS CAN ALSO PROCEED BACKWARDS IN TERMS OF ANY OR ALL INDICES.

IN FORTRAN IV THERE IS ALSO A RULE THAT IF THE TEST VALUE IS ALREADY LESS THAN THE INITIAL VALUE, STILL THE DO LOOP IS EXECUTED ONCE, FOR EXAMPLE, AS IN THE FOLLOWING CASE:-

```
DO 24 J = 25 , 23 , 9
```

IN FORTRAN 77 THE CORRESPONDING RULE IS THAT IN THIS CASE THE DO LOOP IS EXECUTED ZERO TIMES, THAT IS, COMPLETELY SKIPPED. IF WE SET UP A NEST OF SEVERAL DO LOOPS IN FORTRAN 77, AND IF EVEN IN ONE OF THE LOOPS THE TEST VALUE IS LESS THAN THE INITIAL VALUE, THE WHOLE NEST WILL BE SKIPPED. SIMILARLY IN A SITUATION LIKE THE FOLLOWING:-

```
DO 154 M = 15 , 17 , -9
```

(WHICH IS ALSO ALLOWED IN FORTRAN 77 BUT NOT IN FORTRAN IV), THE CALCULATIONS WILL BE SKIPPED.

HOWEVER, I FOUND 2 OR 3 GREAT LACUNAE IN THE RULES OF THE DO STATEMENT. FIRSTLY, WE CAN EASILY CONCEIVE A SITUATION WHERE WE HAVE TO COMPUTE SOME FUNCTION(S) OF MANY VARIABLES, EACH VARIABLE TAKING UP MANY EQUALLY SPACED VALUES INDEPENDENTLY OF THE OTHERS, BUT THE EXACT NUMBER OF SUCH VARIABLES IN NOT KNOWN IN ADVANCE AT THE TIME WHEN WE ARE WRITING THE PROGRAM. IN OTHER WORDS, WE WANT

TO GENERATE A NEST OF N DO LOOPS, WHERE THE VALUE OF N, BEING A NATURAL NUMBER, IS NOT KNOWN AT PROGRAMMING TIME. IN FACT, THE VALUE OF N WILL BE KNOWN ONLY SUBSEQUENTLY AT EXECUTION TIME, I.E., WE SHALL HAVE TO KNOW ITS VALUE THROUGH A READ OR INPUT STATEMENT. NEITHER FORTRAN IV NOR FORTRAN 77 ALLOWS FOR SUCH A FACILITY, BUT IT IS VERY MUCH APPARENT THAT IN REAL LIFE SITUATIONS SUCH A PROBLEM CAN ARISE WHEN WE HAVE TO CALCULATE

$$F(X_1, X_2, X_3, \ldots, X_N)$$

WHERE THE VALUE OF N WILL BE READ BY INPUT AND EACH VARIABLE WILL TAKE UP MANY EQUALLY SPACED VALUES INDEPENDENTLY OF THE OTHERS. WE CAN SET UP THREE 1-DIMENSIONAL ARRAYS A,B,C IN SUCH A SITUATION, WHERE X_1 WILL VARY FROM A_1 TO C_1 IN STEPS OF B_1, X_2 WILL VARY FROM A_2 TO C_2 IN STEPS OF B_2, ETC. SOME OR ALL THE LOOPS MAY PROCEED FORWARDS OR BACKWARDS ACCORDING TO OUR CHOICE. ALSO IN SOME CASES THE TEST VALUE MAY BE LESS THAN THE INITIAL VALUE, THE INCREMENT BEING POSITIVE, OR THE TEST VALUE MAY BE GREATER THAN THE INITIAL VALUE, THE INCREMENT BEING NEGATIVE. HOWEVER IN NO CASE THE INCREMENT CAN BE ZERO. THE INITIAL OR TERMINAL VALUES CAN, HOWEVER, BE POSITIVE, NEGATIVE OR ZERO. THESE VALUES MAY BE INTEGER, REAL OR D.P. ACCORDING TO OUR CHOICE. NOW THIS TYPE OF STRUCTURE IS NOT POSSIBLE TO WRITE IN THE ORDINARY METHOD OF CREATING DO LOOPS, BECAUSE FOR EVERY SEPARATE INDEX WE HAVE TO WRITE A FRESH DO STATEMENT AND WE CAN NEVER WRITE N DO STATEMENTS IN A PROGRAM WITHOUT

//

KNOWING THE VALUE OF N IN ADVANCE.

ANOTHER LACUNA IN THE RULES IS THAT OFTEN IN REAL LIFE SITUATION WE WANT THE INDICES OF ALL THE DO LOOPS TO BE DISTINCT. SOMETIMES WE DEMAND THAT THE INDEX OF AN INNER DO LOOP MUST ALWAYS BE GREATER THAN (LESS THAN) THE INDEX OF AN OUTER DO LOOP. FOR EXAMPLE, IF WE WISH TO PRINT OUT ALL THE PERMUTATIONS OR COMBINATIONS OF THE FIRST N NATURAL NUMBERS, TAKING R AT A TIME, WE CAN DO SO BY A NEST OF R DO LOOPS, BUT WE HAVE TO OBSERVE THESE CONSTRAINTS. IF THE VALUE OF N IS KNOWN IN ADVANCE, SAY N=3, WE CAN WRITE OUR PROGRAM AS FOLLOWS:-

```
      DO 1 I = 1 , 5
      DO 2 J = 1 , 5
      IF ( J .EQ. I ) GO TO 2
      DO 3 K = 1 , 5
      IF ( K .EQ. I ) GO TO 3
      IF ( K .EQ. J ) GO TO 3
      ..............................
      (ACTUAL CALCULATIONS REQUIRED)
      ..............................
3     CONTINUE
2     CONTINUE
1     CONTINUE
```

BUT IF THE VALUE OF N IS LARGE, SAY 50, SUCH A STRUCTURE WILL BE NEXT TO IMPOSSIBLE TO WRITE. IF THE VALUE OF N IS NOT KNOWN AT ALL IN ADVANCE, WE CANNOT EVEN THINK OF THE INDICES I,J,K, BUT WE MUST CREATE AN ARRAY OF INDICES I(1),I(2),I(3),...,I(N). THEN WE CAN MAKE THE COMPARISONS AMONG THESE INDICES BY SETTING UP ANOTHER DO LOOP. IN THAT DO LOOP WE CAN TEST THAT IF J IS NOT EQUAL TO K BUT I(J)=I(K), THEN WE SHALL GO TO THE RELEVANT CONTINUE STATEMENT. NOW THE PROBLEM ARISES: WHERE TO KEEP THIS EXTRA DO LOOP? THE

//

RANGE OF THE ORIGINAL NEST OF DO LOOPS WILL NOT BE THE PROPER PLACE TO KEEP THIS LOOP AND THE CORRECT PLACE IS ONLY OUTSIDE OF IT. THEN THE TROUBLE ARISES THAT UNDER THE RULES IT IS NOT PERMISSIBLE TO ENTER A NEST OF DO LOOPS FROM A POINT OUTSIDE THAT NEST, OR EVEN OUTSIDE SOME OF THE LOOPS. WE CANNOT EVEN THINK OF EXTENDED RANGE OF A DO LOOP, BECAUSE EVEN ACCORDING TO THE RULES OF FORTRAN 77, ONLY THE INNERMOST DO LOOP CAN HAVE AN EXTENDED RANGE. SUPPOSE THAT WE ARE WITHIN 2 OF THE LOOPS BUT OUTSIDE ALL OTHER LOOPS AND WE WISH TO GO TO THE CONTINUE STATEMENT OF THE 2ND DO LOOP. THEN WE CANNOT DO SO FROM THE EXTENDED RANGE. THIS SITUATION WILL ARISE IF WE HAVE ALREADY DEFINED I(1) AND ALSO I(2) AND THEN WE DISCOVER THAT I(2)=I(1); SO WE HAVE TO INCREASE (OR DECREASE) I(2). NOT ONLY FORTRAN, BUT OTHER LANGUAGES ALSO DO NOT PROVIDE FOR SUCH SITUATIONS. THEREFORE IN ONE OF MY PROGRAMS I HAVE TRIED TO RADICALLY EXTEND AND REFINE THE VERY CONCEPT 'NEST OF DO LOOPS'. I HAVE WRITTEN THE PROGRAM ACCORDING TO THE RULES OF FORTRAN IV AND NOT ACCORDING TO THE RULES OF FORTRAN 77. THUS I HAVE CONFINED MYSELF TO ONLY INTEGER INDICES. HOWEVER, I HAVE ALLOWED THE INDICES AND ALL THE 3 INDEXING PARAMETERS TO BECOME POSITIVE, ZERO OR NEGATIVE. FURTHER, IT IS POSSIBLE THAT IN THE NEST SOME LOOPS MAY PROCEED FORWARDS AND SOME MAY PROCEED BACKWARDS. REGARDING THE QUESTION OF THE TERMINAL VALUE ALREADY BEING LESS THAN THE INITIAL VALUE IN CASE THE INCREMENT IS POSITIVE, I HAVE FOLLOWED THE RULES OF FORTRAN IV, I.E., IN SUCH CASES THAT PARTICULAR LOOP WILL BE EXECUTED ONLY ONCE AND NOT ZERO TIMES. SIMILARLY IF THE TERMINAL

// VALUE IS GREATER THAN THE INITIAL VALUE AND THE INCREMENT IS NEGATIVE, THAT PARTICULAR LOOP WILL BE EXECUTED JUST ONCE. WITHIN THE INNERMOST DO LOOP I HAVE PRINTED ALL THE INDICES TO CHECK THAT THE NEST OF LOOPS IS WORKING PROPERLY. IN ORDER TO SHOW THE IMPACT OF THIS PROGRAM ON REAL LIFE SITUATIONS, IT WAS NECESSARY TO THINK *ABOUT SOME FUNCTION OF N VARIABLES, WHERE THE VALUE OF N IS NOT* KNOWN IN ADVANCE BUT KNOWN AT EXECUTION TIME THROUGH INPUT. I COULD THINK OF NO OTHER FUNCTION EXCEPT THE SUM OF ALL THE INDICES, AND SO I HAVE PRINTED THIS SUM ALSO TOGETHER WITH THE INDICES IN THE SAME LINE. I HAVE MADE THE PROGRAMS SUFFICIENTLY GENERAL, SO THAT, IF NEED BE, THE VALUE OF N CAN BECOME EVEN AS SMALL AS 1 IN THE CASE OF N DO LOOPS. I HAVE USED THREE 1-DIMENSIONAL ARRAYS LINV, LTEV AND LCRE IN WHICH I HAVE STORED THE INITIAL VALUES, THE TEST VALUES AND THE INCREMENTS OF THE VARIOUS DO LOOPS. OBVIOUSLY, NONE OF THE ENTRIES OF LCRE CAN BE ZERO AND I HAVE CHECKED THIS FACT THROUGH AN EXTRA DO LOOP. IF SO HAPPENS, I HAVE CAUSED THE COMPUTER TO PRINT 'DATA ILLEGAL' AND ASK FOR FRESH INPUT. I HAVE ALLOWED ALL ENTRIES OF LINV AND LTEV TO BECOME POSITIVE, ZERO OR NEGATIVE.

OFTEN IN REAL LIFE SITUATIONS IT SO HAPPENS THAT WE HAVE A BIG NEST OF MANY DO LOOPS, SAY, 4 LOOPS WHERE I=1(1)10 IN THE FIRST LOOP, J=1(1)15 IN THE SECOND ONE, K=1(1)20 IN THE THIRD ONE AND L=1(1)25 IN THE FOURTH ONE. THEN THE COMPUTER HAS TO MAKE A TOTAL OF 10*15*20*25=75000 SETS OF CALCULATIONS. IT IS VERY MUCH POSSIBLE THAT THE USER OF THE COMPUTER MAY NOT HAVE SO MUCH

COMPUTER TIME AT HIS DISPOSAL AT ONE STRETCH AND HE MIGHT WANT TO USE THE COMPUTER IN DIFFERENT SITTINGS. IN SUCH A CASE, IF HE STARTS THE NEST OF DO LOOPS AGAIN FROM THE SAME POINT ON EVERY OCCASION, HIS PURPOSE WILL NOT BE SERVED BECAUSE EVERY TIME THE COMPUTER WILL PRINT OUT THE SAME SET OF ANSWERS. THIS PROBLEM HAS ACTUALLY ARISEN DURING THE CALCULATIONS OF MY TEACHER WHILE HE WAS CARRYING OUT HIS RESEARCH, BECAUSE HE HAD TO CALCULATE SOME PHYSICAL QUANTITIES WHICH DEPENDED ON MANY PARAMETERS AND IN ONE SITTING HE WAS ALLOWED ONLY 30 MINUTES OF COMPUTER TIME. THE CALCULATIONS OF EACH SET WERE SO LENGTHY AND COMPLICATED THAT THE COMPUTER COULD NOT GIVE OUT MORE THAN 3 OR 4 RESULTS DURING 30 MINUTES. A LACUNA IN THE DO LOOP STRUCTURE IS THAT IT DOES NOT TAKE CARE OF SUCH SITUATIONS. FOR EXAMPLE, CONSIDER THE FOLLOWING NEST OF DO LOOPS:-

```
      DO 10 I=1,10
      DO 11 J=1,15
      DO 12 K=1,20
      ......................
      (ACTUAL CALCULATIONS)
      ......................
12    CONTINUE
11    CONTINUE
10    CONTINUE
```

SUPPOSE THAT DURING THE FIRST SITTING WE HAVE CALCULATED THE RESULTS CORRESPONDING TO (1,1,1),(1,1,2),...(1,1,20),(1,2,1),..., (1,2,5), I.E., 25 RESULTS IN ALL, AND DURING THE NEXT SITTING WE WISH TO START OUR CALCULATIONS FROM (1,2,6). THEN THIS IS NOT POSSIBLE USING THIS STRUCTURE. ONE OF THE COLLEAGUES OF MY TEACHER SUGGESTED THE FOLLOWING TECHNIQUE:-

```
      READ * , II , JJ , KK
      DO 10 I = II , 10
      DO 11 J = JJ , 15
      DO 12 K = KK , 20
      ..............................
      ..............................
      ..............................
12    CONTINUE
11    CONTINUE
10    CONTINUE
```

HE FURTHER TOLD THAT ON EACH SITTING, WE MUST PUNCH AN EXTRA DATA CARD WITH THE VALUES OF II,JJ,KK RELEVANT TO THAT SITTING, AND THEN THE PROGRAM WILL WORK AS DESIRED. FOR EXAMPLE, ON THE FIRST WE CAN PUNCH THE VALUES AS 1,1,1. AFTER OBTAINING 25 RESULTS, WHEN WE WANT TO START FROM (1,2,6), WE CAN PUNCH THE EXTRA CARD WITH 1,2,6 AND SO ON. BUT IF WE DO THAT, THEN AFTER CALCULATING FOR THE SETS (1,2,6),(1,2,7),...(1,2,20), THE COMPUTER WILL START FROM THE SET (1,3,6) AND NOT FROM THE SET (1,3,1) AS WE WOULD NORMALLY WANT IT TO DO. SO A NUMBER OF SETS WILL NOT BE CALCULATED. THUS IT IS CLEAR THAT THIS TECHNIQUE SUGGESTED TO MY TEACHER DURING HIS RESEARCH BY ONE OF HIS COLLEAGUES WAS NOT CORRECT. IT CAN WORK ONLY IF THERE IS A SINGLE DO LOOP, NOT WHEN THERE IS A NEST OF MANY DO LOOPS. THEREFORE IN ONE OF MY PROGRAMS I HAVE TRIED TO INTRODUCE THIS CONDITION ALSO, THAT WE MAY START EXECUTING THE NEST FROM ANY POINT DESIRED, AND THE COMPUTER WILL EXECUTE FOR ALL THE POSSIBLE COMBINATIONS FOR EACH SET IN THE CORRECT ORDER; NEITHER MISSING ANY SET, NOR CALCULATING MORE THAN ONCE FOR ANY SET. FOR THIS PURPOSE I HAVE INTRODUCED AN EXTRA 1-DIMENSIONAL ARRAY CALLED LNET. FOR EXAMPLE, IF IN A PARTICULAR CASE, N=3 AND THE 4 ARRAYS

//

LINV, LTEV, LCRE AND LNET ARE AS FOLLOWS:-

LINV=(3,1,9), LTEV=(6,5,13), LCRE=(1,2,1), LNET=(4,3,11),

THEN THE COMPUTER WILL START CALCULATING FROM THE SET (4,3,11) AND THEN IT WILL CALCULATE FOR (4,3,12),(4,3,13),(4,5,9),(4,5,10), (4,5,11),(4,5,12),(4,5,13),(5,1,9),... UNTIL THE TIME ALLOTTED TO THE USER IS OVER. ON THE NEXT SITTING, THE USER CAN PUNCH A FRESH DATA CARD FOR INPUTTING FRESH VALUES OF LNET AND RE-START THE COMPUTER. FOR EXAMPLE, IF ON THE PREVIOUS SITTING HE HAS COMPUTED UPTO (5,7,10), THEN HE CAN PUNCH LNET AS (5,7,11) AND PROCEED. IN THE FIRST PROGRAM, GIVEN ON THE SUCCEEDING PAGES, WE HAVE INTRODUCED THE FEATURE THAT WE CAN RUN A NEST OF N DO LOOPS, N NOT BEING KNOWN IN ADVANCE, WITH ALL POSSIBILITIES, NAMELY, SOME LOOPS MAY GO FORWARDS AND SOME BACKWARDS, AND THE STARTING VALUES FROM WHERE WE WANT TO PROCEED HAVE TO BE READ IN SEPARATELY. IN OUR SECOND PROGRAM WE HAVE CONSIDERED THE CONDITION OF ALL INDICES BEING DISTINCT, BUT WE HAVE NOT ALLOWED FOR STARTING FROM ANY GIVEN SET OF VALUES. OF COURSE, IF WE LIKE, WE CAN JOIN THE 2 PROGRAMS TO ALLOW FOR BOTH SITUATIONS. IN THE SECOND PROGRAM, IT IS ALSO POSSIBLE THAT THE CONDITION FOR DISTINCT INDICES MAY BECOME IMPOSSIBLE. FOR EXAMPLE, IN THE FOLLOWING CASE:-

```
      DO 24 I=1,3
      DO 24 J=1,3
      DO 24 K=1,3
      DO 24 L=1,3
      ..................
24    CONTINUE
```

IF WE MAKE THE RESTRICTION OF DISTINCT INDICES, NO SET IS POSSIBLE. SO IN SUCH A CASE, THE PROGRAM PRINTS OUT AN APPROPRIATE MESSAGE AND DOES NOT PERFORM ANY CALCULATION. BOTH PROGRAMS ARE GIVEN ON THE SUCCEEDING PAGES.

2. PROGRAM FOR CREATING A NEST OF N DO LOOPS which ~~AND~~ STARTS THE EXECUTION FROM ANY SET OF STARTING VALUES.

```
      DIMENSION LINV ( 25 ) ,LTEV ( 25 ) ,LCRE ( 25 ) ,LNET ( 25 ) ,
     1 LDEX ( 25 )
      WRITE ( 1 , 7 )
7     FORMAT ( 1X , 'THIS PROGRAM CREATES A NEST OF N DO LOOPS AND',
     1 1X , 'STARTS THE EXECU"ION FROM' / 1X , 'ANY SET OF STARTING',
     2 1X , 'VALUES' )
19    WRITE ( 1 , 27 )
27    FORMAT ( 1X , 'TYPE THE NUMBER OF DO LOOPS IN FORMAT ( I2 )'
     1 / 1X , 'FOR STOPPING : PLEASE TYPE 0' )
191   READ ( 1 , 47 ) ND
47    FORMAT( I2 )
      IF ( ND ) 79 , 799 , 152
79    WRITE ( 1 , 127 )
127   FORMAT ( 1X , 'DATA ILLEGAL, KINDLY TYPE  AGAIN' )
      GO TO 191
152   IF ( ND .GT. 25 ) GO TO 749
      IT = 1
189   WRITE ( 1 , 197 )
197   FORMAT ( 1X , 'TYPE INITIAL VALUES, TEST VALUES AND INCREMENTS',
     1 1X , 'OF EACH DO LOOP IN' / 1X , 'FORMAT ( 3I2 ) IN SEPARATE',
     2 1X , 'LINES, THERE BEING 1 LINE FOR EACH LOOP' )
1891  DO 222 J = IT , ND
      READ ( 1  , 207 ) LINV ( J ) , LTEV ( J ) , LCRE ( J )
207   FORMAT ( 3I2 )
C     INCREMENT ZERO IS NOT POSSIBLE
      IF ( LCRE ( J ) .EQ. 0 ) GO TO 675
222   CONTINUE
      WRITE ( 1 , 227 )
227   FORMAT ( 1X , 'TYPE ALL STARTING VALUES OF THE INDICES IN',
     1 1X , 'FORMAT ( 25I2 )' )
1000  READ ( 1 , 237 ) ( LNET ( J ) , J = 1 , ND )
237   FORMAT ( 25I2 )
C     VALIDATE THE DATA
      DO 355 J = 1 , ND
      IF ( LINV ( J ) - LNET ( J ) ) 253 , 355 , 286
253   IF ( LCRE ( J ) .LT. 0 .OR. LNET ( J ) .GT. LTEV ( J ))GO TO 791
      GO TO 318
286   IF ( LCRE ( J ) .GT. 0 .OR. LNET ( J ) .LT. LTEV ( J ))GO TO 791
318   IF ( MOD ( LNET ( J ) - LINV ( J ) , LCRE ( J )) .NE.0)GO TO 791
355   CONTINUE
```

```
C       PUT INDICES OF ALL DO LOOPS EQUAL TO THE STARTING VALUES
        DO 388  K = 1  , ND
388     LDEX ( K ) = LNET ( K )
415     KP = ND
450     NSUM = 0
        DO 500 I = 1  , ND
500     NSUM = NSUM + LDEX ( I )
        WRITE ( 1  , 489 ) ( LDEX ( K ) , K = 1  , ND )  , NSUM
489     FORMAT ( 1X , 2014 )
C       CHANGE THE INDEX OF A DO LOOP
        DO 644 J = 1  , ND
        LDEX ( KP ) = LDEX ( KP ) + LCRE ( KP )
C       TEST WHETHER THE PARTICULAR INCREMENT IS POSITIVE OR NEGATIVE
        IF ( LCRE ( KP ) ) 520 , 799 , 582
520     IF ( LDEX ( KP ) .GE. LTEV ( KP ) ) GO TO 552
        GO TO 614
552     IF ( KP - ND ) 415 , 450 , 799
582     IF ( LDEX ( KP ) .LE. LTEV ( KP ) ) GO TO 552
C       IF CHANGE IN INDEX OF A PARTICULAR DO LOOP IS NOT POSSIBLE, SET
C       THE INDEX OF THAT DO LOOP EQUAL TO THE INITIAL VALUE
614     LDEX ( KP ) = LINV ( KP )
644     KP = KP -1
        GO TO 19
675     IT = J
        WRITE ( 1  , 707 )
707     FORMAT ( 1X ,'THIS DATA SET IS ILLEGAL, KINDLY TYPE IT AGAIN' )
        GO TO 1891
749     WRITE ( 1  , 787 )
787     FORMAT ( 1X , 'THIS NUMBER WILL CROSS THE DIMENSION, CHANGE',
     1  1X , 'DIMENSION' )
799     STOP
791     WRITE ( 1  , 2406 )
2406    FORMAT ( 1X , 'SOME OR ALL STARTING VALUES ARE WRONG,'
     1  1X , 'PLEASE TYPE THEM AGAIN' )
        GO TO 1000
        END
```

3. PROGRAM FOR CREATING A NEST OF N DO LOOPS WITH THE CONDITION THAT THE INDICES OF ALL THE DO LOOPS ARE DISTINCT.

```
      DIMENSION LINV ( 20 ) , LTEV ( 20 ) , LCRE ( 20 ) , LDEX ( 20 )
      WRITE ( 1 , 7 )
7     FORMAT ( 1X , 'THIS PROGRAM CREATES A NEST OF N DO LOOPS WITH',
     1 1X , 'THE CONDITION THAT' / 1X , 'THE INDICES OF ALL THE DO',
     2 1X , 'LOOPS ARE DISTINCT' )
29    WRITE ( 1 , 17 )
17    FORMAT ( 1X , 'TYPE THE NUMBER OF DO LOOPS IN FORMAT ( I2 )' )
2     READ ( 1 , 47 ) ND
47    FORMAT ( I2 )
      IF ( ND ) 89 , 999 , 165
89    WRITE ( 1, 127 )
127   FORMAT ( 1X , 'DATA ILLEGAL, KINDLY TYPE IT AGAIN' )
      GO TO 2
165   IF ( N .GT. 20 ) GO  TO 909
      IT = 1
      WRITE ( 1 , 177 )
177   FORMAT ( 1X , 'TYPE INITIAL VALUES, TEST VALUES, INCREMENTS'
     1 / 1X , 'OF DO LOOPS IN FORMAT (3I2), ONE SET PER DATA CARD' )
180   DO 211 J = IT , ND
      READ ( 1 , 197 ) LINV ( J ) , LTEV ( J ) , LCRE ( J )
197   FORMAT ( 3I2 )
      IF ( LCRE ( J ) .EQ. 0 ) GO TO 676
211   CONTINUE
      IF ( ND .EQ. 1 ) GO TO 738
      LP = 0
      INDEX =  0
C     SET THE INDEX OF EACH DO LOOP TO ITS INITIAL VALUE
      DO 255 J = 1 , ND
255   LDEX ( J ) = LINV ( J )
      LP = 2
286   IF ( LP .EQ. 1 ) LP = 2
      DO 344 K = LP , ND
      IPV = LDEX ( K )
      MN1 = K - 1
C     COMPARE THE INDICES OF THE DO LOOPS
      DO 311 J = 1 , MN1
      IF ( LDEX ( J ) .EQ. IPV ) GO TO 480
311   CONTINUE
C     IF DISTINCT INDICES EXIST, THEN INDEX=1, OTHERWISE INDEX=0
344   CONTINUE
      IF ( INDEX .EQ. 1 ) GO TO 375
```

```
      INDEX = 1
375   NSUM = 0
      DO 236 I = 1 , ND
236   NSUM = NSUM + LDEX ( I )
      WRITE ( 1 , 407 ) ( LDEX ( K ) , K = 1 , ND ) , NSUM
407   FORMAT ( 1X , 16I5 )
      IF ( LP - ND ) 448 , 351 , 999
448   LP = ND
      GO TO 516
480   LP = K
516   M = LP
C     CHECK WHETHER A PARTICULAR INDEX CAN BE INCREASED OR DECREASED
551   DO 644 J = 1 , M
      LDEX ( LP ) = LDEX ( LP ) + LCRE ( LP )
      IF ( LCRE ( LP ) ) 582 , 999 , 614
582   IF ( LDEX ( LP ) .GE. LTEV ( LP ) ) GO TO 286
      GO TO 632
614   IF ( LDEX ( LP ) .LE. LTEV ( LP ) ) GO TO 286
632   LDEX ( LP ) = LINV ( LP )
644   LP = LP -1
      GO TO 853
676   IT = J
      WRITE ( 1 , 707 )
707   FORMAT ( 1X , 'NO INCREMENT CAN BE ZERO, KINDLY TYPE THIS SET',
     1 1X , 'AGAIN' )
      GO TO 180
C     IF THERE IS ONLY 1 DO LOOP, THEN ALL THE POSSIBLE VALUES OF THE
C     INDEX MUST BE CONSIDERED
738   K1 = LINV ( 1 )
      K2 = LTEV ( 1 )
      K3 = LCRE ( 1 )
769   WRITE ( 1 , 407 ) K1 , K1
      K1 = K1 + K3
      IF ( K3 ) 801 , 999 , 832
801   IF ( K1 .GE. K2 ) GO TO 769
      GO TO 29
832   IF ( K1 .LE. K2 ) GO TO 769
      GO TO 29
853   IF (INDEX .EQ. 1 ) GO TO 29
      WRITE ( 1 , 887 )
887   FORMAT ( 1X , 'NO SET OF DISTINCT VALUES EXISTS')
      GO TO 29
909   WRITE ( 1 , 947 )
947   FORMAT ( 1X , 'THIS NUMBER WILL CROSS THE DIMENSION,',
     1 1X , 'CHANGE DIMENSION' )
999   STOP
      END
```

//

AS A BY-PRODUCT OF THESE PROGRAMS, WE ALSO PRESENT A PROGRAM WHICH PRINTS OUT ALL THE PERMUTATIONS OF N NATURAL NUMBERS TAKEN R AT A TIME. WE ALSO PRESENT A PROGRAM WHICH PRINTS OUT ALL THE COMBINATIONS OF N NATURAL NUMERS TAKEN R AT A TIME. THE LATTER PROGRAM IS ALSO USEFUL IN THIS RESPECT THAT IN THE NEXT CHAPTER WHEN WE SHALL COMPUTE THE CHARACTERISTIC POLYNOMIAL OF A SQUARE MATRIX, WE SHALL NEED ALL COMBINATIONS OF THE FIRST N NATURAL NUMBERS TAKEN R AT A TIME. AT THAT TIME WE SHALL USE THIS PROGRAM IN THE CHARACTERISTIC POLYNOMIAL PROGRAM. THESE 2 PROGRAMS ARE GIVEN ON THE SUCCEEDING PAGES.

//

4. PROGRAM FOR GENERATING ALL POSSIBLE PERMUTATIONS OF THE FIRST N NATURAL NUMBERS TAKING R AT A TIME.

```
      DIMENSION LDEX ( 20 )
      WRITE ( 1 , 7 )
7     FORMAT ( 1X , 'THIS PROGRAM PRINTS ALL PERMUTATIONS OF THE',
     1 1X , 'FIRST N NATURAL NUMBERS TAKING IR' / 1X , 'AT A TIME',
     2 1X , 'AND THE TOTAL NUMBER OF PERMUTATIONS' )
9     WRITE ( 1 , 17 )
17    FORMAT ( 1X , 'TYPE N AND IR IN FORMAT ( 2I2 )' / 1X ,
     1 'FOR STOPPING : PLEASE TYPE 0' )
191   READ ( 1 , 47 ) N , IR
47    FORMAT ( 2I2 )
      IF ( N ) 89 , 599 , 145
89    WRITE ( 1, 117 )
117   FORMAT ( 1X , 'DATA ILLEGAL, KINDLY TYPE AGAIN' )
      GO TO 191
145   IF ( N .GT. 20 ) GO TO 509
      IF ( IR .LE. 0 .OR. IR .GT. N ) GO TO 89
      WRITE ( 1 , 149 ) N , IR
      NP = 0
149   FORMAT ( 1X , 'PERMUTATIONS OF' , 1X , I2 , 1X , 'TAKING' , 1X ,
     1 I2 , 1X , 'AT A TIME' , 1X , 'ARE GIVEN BELOW:-' )
      IF ( IR .EQ. 1 ) GO TO 442
      LP = 0
      DO 177 J = 1 , IR
177   LDEX ( J ) = J
      GO TO 309
208   IF ( LP .EQ. 1 ) LP = 2
      DO 266 K = LP , IR
      IPV = LDEX ( K )
      MN1 = K - 1
      DO 244 J = 1 , MN1
      IF ( LDEX ( J ) .EQ. IPV ) GO TO 402
244   CONTINUE
266   CONTINUE
309   WRITE ( 1 , 337 ) ( LDEX ( K ) , K = 1 , IR )
337   FORMAT ( 1X , 16I5 )
      NP = NP + 1
      IF ( LP - IR ) 368 , 475 , 599
368   LP = IR
      GO TO 434
402   LP = K
```

```
434      M = LP
475      DO 411 J = 1 , M
         LDEX ( LP ) = LDEX ( LP ) + 1
         IF ( LDEX ( LP ) .LE. N ) GO TO 208
         LDEX ( LP ) = 1
411      LP = LP - 1
419      WRITE ( 1 , 427 ) NP
427      FORMAT ( 1X , 'TOTAL NUMBER OF PERMUTATIONS =' , 1X , I4 )
         GO TO 9
442      K1 = 1
479      WRITE ( 1 , 337 ) K1
         NP = NP + 1
         K1 = K1 + 1
         IF ( K1 .LE. N ) GO TO 479
         GO TO 419
509      WRITE ( 1 , 547 )
547      FORMAT ( 1X, 'THE GIVEN NUMBER WILL CROSS THE DIMENSION,', 1X ,
        1 'CHANGE DIMENSION' )
599      STOP
         END

//
```

5. PROGRAM FOR GENERATING ALL POSSIBLE COMBINATIONS OF THE FIRST N NATURAL NUMBERS TAKING R AT A TIME.

```
      DIMENSION LDEX ( 25 )
      INTEGER R
      WRITE ( 1 , 47 )
47    FORMAT ( 1X , 'THIS PROGRAM PRINTS ALL COMBINATIONS OF THE',
     1  , 1X , 'FIRST N NATURAL NUMBERS TAKING R' / 1X , 'AT A TIME',
     1  , 1X , 'AND THE TOTAL NUMBER OF COMBINATIONS' )
79    WRITE ( 1 , 107 )
107   FORMAT ( 1X , 'TYPE N AND R IN FORMAT ( 2I2 )'
     1  / 1X , 'FOR STOPPING :PLEASE TYPE 0' )
791   READ ( 1 , 127)  N , R
127   FORMAT ( 2I2 )
      IF ( N ) 149 , 499 , 192
149   WRITE ( 1 , 167 )
167   FORMAT ( 1X, 'DATA ILLEGAL, PLEASE TYPE AGAIN'  )
      GO TO 791
192   IF ( N .GT. 25 ) GO TO 419
      IND = 1
201   CALL COMB ( N , R , IND , LDEX , K , IEV , KP   )
      GO TO ( 149 , 249 , 319 ) , IND
249   IF ( K .GT. 1 ) GO TO 279
      WRITE ( 1 , 267 ) N , R
267   FORMAT ( 1X , 'COMBINATIONS OF THE FIRST' , 1X , I2 , 1X,
     1  'NATURAL NUMBERS TAKING' , 1X , I2 , 1X , 'AT A TIME' / 1X ,
     2  'ARE GIVEN BELOW:-' )
279   WRITE ( 1 , 287 ) ( LDEX ( KK ) , KK = 1 , R )
287   FORMAT ( 1X , 25 ( I2 , 1X ) )
      GO TO 201
319   WRITE ( 1 , 357 ) K
357   FORMAT ( 1X , 'TOTAL NUMBER OF COMBINATIONS =' , 1X , I4 )
      GO TO 79
419   WRITE ( 1 , 467 )
467   FORMAT ( 1X, 'THE GIVEN NUMBER WILL CROSS THE DIMENSION,'
     1  ,1X , 'CHANGE DIMENSION' )
499   STOP
      END
     SUBROUTINE COMB ( N , R , IND , LDEX , K , IEV , KP )
     DIMENSION LDEX ( 25 )
     INTEGER R
     IF ( IND .NE. 1 ) GO TO 515
     IF (  R .GT. 0 .AND. N .GE. R ) GO TO 242
```

```
202     RETURN
242     IND = 2
        IEV = 0
        K = 0
        DO 288 I = 1 , R
288     LDEX ( I ) = I
        GO TO 450
300     IF ( KP - R ) 360 , 540 , 202
360     IF ( IEV .EQ. LDEX ( KP ) ) GO TO 392
        IEV = LDEX ( KP )
392     DO 411 I = KP , R
        IEV = IEV + 1
411     LDEX ( I ) = IEV
450     KP = R
485     K = K + 1
        RETURN
515     IF ( N + KP - R - ( LDEX ( KP ) + 1 ) ) 575 , 540 , 300
540     LDEX ( KP ) = LDEX ( KP ) + 1
        GO TO 485
575     IF ( KP .EQ. 1 ) GO TO 700
        KP = KP - 1
        GO TO 515
700     IND = 3
        RETURN
        END

//
```

//

CHAPTER 4

SOME PROGRAMS REGARDING SQUARE MATRICES

1. INTRODUCTION

IN THIS CHAPTER WE SHALL DEVELOP A FEW PROGRAMS REGARDING MANIPULATION OF SQUARE MATRICES. AS IN THE EARLIER CHAPTERS, WE SHALL KEEP OUR INVESTIGATIONS LIMITED TO THE CASE WHEN ALL THE ENTRIES OF THE MATRICES ARE INTEGERS. OF COURSE, WE CAN EASILY EXTEND OUR PROGRAMS TO THE CASE WHEN THE ENTRIES OF THE MATRICES ARE NOT NECESSARILY INTEGERS, BUT CAN BE RATIONAL NUMBERS, NOT NECESSARILY IN THEIR LOWEST TERMS, BUT WITH THE CONDITION THAT THE DENOMINATORS OF THOSE ENTRIES WILL NEVER BE ZERO. FOR THIS PURPOSE WE CAN USE THE TECHNIQUES DEVELOPED IN CHAPTER 2. OUR FIRST PROGRAM GENERATES THE CHARACTERISTIC POLYNOMIAL OF A SQUARE MATRIX OF ORDER N, WHERE THE VALUE OF N IS NOT KNOWN IN ADVANCE, BUT IS GIVEN TO THE PROGRAM BY MEANS OF A READ STATEMENT. SINCE ALL ENTRIES OF THE MATRIX ARE INTEGERS, IT FOLLOWS THAT THE CHARACTERISTIC POLYNOMIAL WILL BE MONIC AND ALL ITS COEFFICIENTS WILL BE INTEGERS. AS IN CHAPTER 2, WE PRINT ALL COEFFICIENTS EXCEPT THE LEADING COEFFICIENT. IF THE ENTRIES WOULD HAVE BEEN RATIONAL NUMBERS, THEN IN STANDARD FORM THE CHARACTERISTIC POLYNOMIAL WOULD NOT HAVE BEEN NECESSARILY MONIC, BUT ALL ITS COEFFICIENTS WOULD STILL HAVE BEEN INTEGERS AND THEIR H.C.F. WOULD HAVE BEEN UNITY. FURTHER, THE LEADING COEFFICIENT CAN BE MADE POSITIVE IN THIS CASE. THIS PROGRAM CAN EASILY BE WRITTEN IN A VERY SIMILAR FASHION. HOWEVER, WE HAVE NOT WRITTEN IT HERE.

//

OUR SECOND PROGRAM COMPUTES ALL THE INTEGER CHARACTERISTIC ROOTS OF A SQUARE MATRIX WITH INTEGER ENTRIES. IN CASE ONLY SOME OF THE CHARACTERISTIC ROOTS ARE INTEGERS, IT PRINTS THEM TOGETHER WITH AN APPROPRIATE MESSAGE. IN CASE NONE OF THE CHARACTERISTIC ROOTS ARE INTEGERS, IT PRINTS OUT ONLY A MESSAGE TO THIS EFFECT.

THE MAIN DIFFICULTY WITH OUR SECOND PROGRAM IS THAT IF THE ORDER OF A SQUARE MATRIX HAPPENS TO BE GREATER THAN 2, IT IS VERY DIFFICULT (ALMOST IMPOSSIBLE) TO WRITE DOWN ANY NON-TRIVIAL MATRIX WHOSE CHARACTERISTIC ROOTS ARE INTEGERS. IF ONE WRITES ANY MATRIX, EVEN OF VERY SMALL ORDER, SAY 3 OR 4, WITH ARBITRARY INTEGER ENTRIES, AND COMPUTES ITS CHARACTERISTIC POLYNOMIAL, ONE ALMOST INVARIABLY FINDS THAT NONE OF ITS ZEROS ARE INTEGERS. SINCE THE POLYNOMIAL IS MONIC, WE CAN EVEN SAY THAT NONE OF ITS ZEROS ARE RATIONAL. THEREFORE, IT BECOMES VERY DIFFICULT EVEN TO TEST THIS PROGRAM. IN ORDER TO GET SAMPLE DATA FOR THIS PROGRAM WE HAVE USED A SPECIAL TECHNIQUE. WE KNOW THAT THE CHARACTERISTIC ROOTS OF A DIAGONAL MATRIX D ARE JUST ITS DIAGONAL ENTRIES. WE ALSO KNOW THAT IF P IS ANY NON-SINGULAR MATRIX OF ORDER N AND D IS ANY DIAGONAL MATRIX OF THE SAME ORDER, THEN IF WE DEFINE $A=PDP^{-1}$, THEN THE MATRICES D AND A ARE CALLED SIMILAR. ALSO WE KNOW THAT SIMILAR MATRICES HAVE THE SAME CHARACTERISTIC ROOTS. THEREFORE, IF WE START WITH ANY DIAGONAL MATRIX WITH INTEGER ENTRIES, AND PRE-MULTIPLY THIS MATRIX WITH ANY NON-SINGULAR MATRIX P, AND POST-MULTIPLY IT BY P^{-1}, THEN WE ARE SURE TO GET A NON-TRIVIAL MATRIX A WHOSE CHARACTERISTIC ROOTS WILL BE THE SAME AS THOSE OF D, I.E.,

//

INTEGERS. SO WE CAN TEST OUR CHARACTERISTIC ROOTS PROGRAM BY USING THIS MATRIX A.

HOWEVER, THERE REMAINS ONE MORE DIFFICULTY. IF WE WRITE ANY SQUARE MATRIX P WITH ARBITRARY INTEGER ENTRIES, IT IS NOT EASY TO ENSURE THAT ITS DETERMINANT WILL BE UNITY. IF ITS DETERMINANT IS NOT +1 OR -1, EVEN IF IT IS NON-SINGULAR, THE DETERMINANT OF ITS INVERSE MATRIX WILL NOT BE AN INTEGER AND SO THE ENTRIES OF THE INVERSE MATRIX WILL NOT ALL BE INTEGERS. SO OUR PURPOSE IS DEFEATED. THEREFORE WE HAVE DEVELOPED ONE MORE PROGRAM IN THIS CHAPTER, WHICH GENERATES A SQUARE MATRIX OF ORDER N, ALL ITS ENTRIES BEING INTEGERS, AND WITH THE CONDITION THAT ITS DETERMINANT IS ALWAYS UNITY, SO THAT ITS INVERSE MATRIX WILL ALSO HAVE ALL INTEGER ENTRIES. USING THIS PROGRAM WE CAN EASILY GET THE NECESSARY DATA FOR TESTING THE CHARACTERISTIC ROOTS PROGRAM.

2. ALGORITHM FOR CALCULATING THE CHARACTERISTIC POLYNOMIAL .

LET A BE A SQUARE MATRIX OF ORDER N, ALL OF WHOSE ENTRIES ARE INTEGERS. WE CONSIDER THE FORMAL MATRIX POLYNOMIAL A-XI, WHERE X IS AN INDETERMINATE AND I IS THE UNIT MATRIX OF ORDER N. THIS MATRIX POLYNOMIAL WILL ITSELF BE A SQUARE MATRIX OF ORDER N, WHOSE ENTRIES WILL BE FORMAL POLYNOMIALS IN THE INDETERMINATE X. FOR EXAMPLE, IF N=3 AND IF $A = \begin{bmatrix} -2 & 0 & 3 \\ 1 & -5 & 6 \\ 0 & 9 & 4 \end{bmatrix}$,

THEN A-XI WILL BECOME $\begin{bmatrix} -2-X & 0 & 3 \\ 1 & -5-X & 6 \\ 0 & 9 & 4-X \end{bmatrix}$.

THE DETERMINANT OF A-XI, I.E., DET(A-XI) OR $|A-XI|$ IS CALLED THE CHARACTERISTIC POLYNOMIAL OF A AND ITS ZEROS ARE CALLED THE

//

CHARACTERISTIC ROOTS OF A. IN THE ABOVE NUMERICAL EXAMPLE, THE CHARACTERISTIC POLYNOMIAL OF A WILL BE

$$\begin{vmatrix} -2-X & 0 & 3 \\ 1 & -5-X & 6 \\ 0 & 9 & 4-X \end{vmatrix}$$

TO MAKE THE ABOVE POLYNOMIAL MONIC, WE CONSIDER $|XI-A|$ INSTEAD OF $|A-XI|$, WHERE $|XI-A| = \begin{vmatrix} X+2 & 0 & -3 \\ -1 & X+5 & -6 \\ 0 & -9 & X-4 \end{vmatrix}$

IN GENERAL, $|XI-A|$ CAN BE BROKEN UP INTO THE SUM OF 2^N DETERMINANTS. IN THE PRESENT CASE WE HAVE $|XI-A|$

$$= \begin{vmatrix} X & 0 & 0 \\ 0 & X & 0 \\ 0 & 0 & X \end{vmatrix} + \begin{vmatrix} 2 & 0 & 0 \\ -1 & X & 0 \\ 0 & 0 & X \end{vmatrix} + \begin{vmatrix} X & 0 & 0 \\ 0 & 5 & 0 \\ 0 & -9 & X \end{vmatrix} + \begin{vmatrix} X & 0 & -3 \\ 0 & X & -6 \\ 0 & 0 & -4 \end{vmatrix}$$

$$+ \begin{vmatrix} 2 & 0 & 0 \\ -1 & 5 & 0 \\ 0 & -9 & X \end{vmatrix} + \begin{vmatrix} 2 & 0 & -3 \\ -1 & X & -6 \\ 0 & 0 & -4 \end{vmatrix} + \begin{vmatrix} X & 0 & -3 \\ 0 & 5 & -6 \\ 0 & -9 & -4 \end{vmatrix} + \begin{vmatrix} 2 & 0 & -3 \\ -1 & 5 & -6 \\ 0 & -9 & -4 \end{vmatrix}$$

OUT OF THESE 2^N DETERMINANTS WE SHALL HAVE $\binom{N}{0}$, I.E., 1 DETERMINANT WHICH WILL CONTAIN NO ENTRIES OF A, $\binom{N}{1}$ DETERMINANTS WHICH WILL CONTAIN 1 COLUMN OF A WITH ALL SIGNS CHANGED, $\binom{N}{2}$ DETERMINANTS WHICH WILL CONTAIN 2 COLUMNS OF A WITH ALL SIGNS CHANGED, AND SO ON. THERE WILL BE $\binom{N}{N}$, I.E., 1 DETERMINANT WHICH WILL BE JUST THE DETERMINANT OF -A. SO IF WE EXPRESS THE CHARACTERISTIC POLYNOMIAL OF A IN THE FORM

$$X^N + C_1 X^{N-1} + C_2 X^{N-2} + \ldots + C_{N-1} X + C_N ,$$

THEN IN ORDER TO CALCULATE $C_1, C_2, C_3, \ldots, C_N$, WE HAVE TO REPLACE X BY 1 IN EVERY DETERMINANT. THEREFORE IN THE COMPUTER PROGRAM WHICH CALCULATES $C_1, C_2, C_3, \ldots, C_N$, WE GENERATE ALL POSSIBLE DETERMINANTS TAKING 1 COLUMN FROM A (SIGNS CHANGED), ALL POSSIBLE DETERMINANTS

TAKING 2 COLUMNS FROM A (SIGNS CHANGED), AND SO ON. WE FILL UP THE REMAINING COLUMNS OF EACH DETERMINANT BY STRINGS OF 0'S WITH EXACTLY ONE 1 AT THE PROPER PLACE. THEN WE EVALUATE EACH DETERMINANT USING A SEPARATE FUNCTION SUBPROGRAM AND ADD THE DETERMINANTS OF EACH KIND TO GET $C_1, C_2, C_3, \ldots, C_N$. WE SHALL ALSO NEED A SUBROUTINE SUBPROGRAM TO GENERATE ALL POSSIBLE COMBINATIONS OF THE FIRST N NATURAL NUMBERS TAKING 1 AT A TIME, TAKING 2 AT A TIME, ETC. WE TAKE THIS PROGRAM FROM THE PREVIOUS CHAPTER AND CONVERT IT INTO A SUBROUTINE SUBPROGRAM. THE PROGRAM FOR EVALUATING A DETERMINANT, CALLED LDET, HAS BEEN WRITTEN BY MY TEACHER. HE HAS EVALUATED THE DETERMINANT BY TRIANGULATING IT BY THE METHOD OF ELEMENTARY ROW TRANSFORMATIONS. DURING THESE TRANSFORMATIONS WE COME ACROSS VULGAR FRACTIONS, BECAUSE, SAY, IF THE 2ND ROW STARTS FROM THE ENTRY C_{21} AND THE 1ST ROW STARTS FROM THE ENTRY C_{11}, THEN WE HAVE TO SUBTRACT C_{21}/C_{11} TIMES THE 1ST ROW FROM THE 2ND ROW. C_{21}/C_{11} MAY NOT BE NECESSARILY AN INTEGER. THEREFORE THE FUNCTION SUBPROGRAM CALLED LDET TAKES THE HELP OF SPECIAL SUBROUTINES WHICH SUBTRACT ONE VULGAR FRACTION FROM ANOTHER, MULTIPLY ONE VULGAR FRACTION BY ANOTHER, AND COMPUTE THE RECIPROCAL OF A VULGAR FRACTION. MY TEACHER HAS WRITTEN THESE SUBROUTINES ALSO AND I HAVE USED THEM IN MY PROGRAM. IT MUST BE MENTIONED THAT IN COMPUTER PROGRAMMING THERE IS NO WAY TO STORE A VULGAR FRACTION AS SUCH IN THE MEMORY; WE MUST STORE THE NUMERATOR AND DENOMINATOR IN SEPARATE MEMORY LOCATIONS. ALSO, TO AVOID STORING UNNECESSARILY BIG NUMBERS, WE DIVIDE THE NUMERATOR AND THE DENOMINATOR OF EACH FRACTION

BY THEIR H.C.F. AFTER EVERY SINGLE COMPUTATION. THEREFORE WE ALSO NEED AN ADDITIONAL FUNCTION SUBPROGRAM CALLED HCF(M,N).

THE COMPLETE PROGRAM FOR CALCULATING THE COEFFICIENTS OF THE CHARACTERISTIC POLYNOMIAL, TOGETHER WITH ALL THE SUBPROGRAMS USED, IS GIVEN ON THE SUCCEEDING PAGES.

```
      DIMENSION MAT ( 10 , 10 ) , N ( 10 ) , NMAT ( 10 , 10 ) ,
     1LDEX ( 10 )
      COMMON / L1 / M
      WRITE ( 1 , 7 )
7     FORMAT ( 1X , 'THIS PROGRAM GENERATES THE CHARACTERISTIC' , 1X,
     1'POLYNOMIAL'/1X,'OF A GIVEN SQUARE MATRIX WITH INTEGER ENTRIES')
79    WRITE ( 1 , 107 )
107   FORMAT ( 1X , 'TYPE THE ORDER OF THE MATRIX IN FORMAT ( I2 )'
     1/1X , 'FOR STOPPING PLEASE TYPE : 0' )
791   READ ( 1 , 127 ) M
127   FORMAT ( I2 )
      IF ( M ) 149 , 799 , 198
149   WRITE ( 1 , 167 )
167   FORMAT (1X , 'DATA ILLEGAL, KINDLY TYPE AGAIN ' )
      GO TO 791
198   IF ( M .GT. 10 ) GO TO 749
      WRITE ( 1 , 227 )
227   FORMAT ( 1X ,'TYPE THE ENTRIES OF THE MATRIX IN FORMAT ( 13I6 )')
      READ ( 1 , 257 ) ( ( MAT ( I , JJ ) , JJ = 1 , M ) , I = 1 , M )
257   FORMAT ( 13I6 )
      DO 488 I = 1 , M
      N ( I ) = 0
      IF ( I .EQ. M ) GO TO 420
      IND = 0
291   CALL COMB (  I , IND , LDEX , IEV , KP  )
      GO TO ( 234 , 488 ) , IND
234   DO 411 KK = 1 , M
      DO 411 LL = 1 , M
      IF ( LL .EQ. KK ) GO TO 410
      NMAT ( LL , KK ) = 0
      GO TO 411
410   NMAT ( LL , KK ) = 1
411   CONTINUE
      DO 713 K = 1 , I
      LDEXK = LDEX ( K )
      DO 713 LL = 1 , M
713   NMAT ( LL , LDEXK ) = - MAT ( LL , LDEXK )
      N ( I ) = N ( I ) + LDET ( NMAT )
      GO TO 291
420   DO 422 J = 1 , M
      DO 422 LL = 1 , M
422   NMAT ( LL , J ) = - MAT ( LL , J )
      N ( I ) = N ( I ) + LDET ( NMAT )
488   CONTINUE
      WRITE ( 1 , 497 )
497   FORMAT ( 1X , 'THE GIVEN MATRIX IS AS FOLLOWS' )
      DO 511 MM = 1 , M
511   WRITE ( 1 , 587 ) ( MAT ( MM , KK ) , KK = 1 , M )
      WRITE ( 1 , 527 )
527   FORMAT ( 1X , 'THE REQUIRED POLYNOMIAL IS MONIC; SO ITS FIRST',
```


```
     11X, 'COEFFICIENT IS UNITY' / 1X, 'ITS OTHER COFFICIENTS,' , 1X,
     1'STARTING FROM THE SECOND, ARE GIVEN BELOW:-' )
      WRITE ( 1 , 587 ) ( N ( I ) , I = 1 , M )
587   FORMAT ( 1X , 11 ( I6 , 1X ) )
      GO TO 79
749   WRITE ( 1 , 787 )
787   FORMAT ( 1X , 'THE GIVEN NUMBER WILL CROSS THE DIMENSION,' , 1X
     1,'CHANGE DIMENSION' )
799   STOP
      END
       SUBROUTINE COMB (  N , IND , LDEX , IEV , KP  )
       DIMENSION LDEX(10)
       COMMON / L1 / M
       IF ( IND .NE. 0 ) GO TO 515
       IND = 1
       IEV = 0
       DO 222 I = 1 , N
222    LDEX(I)=I
       GO TO 450
300    IF ( KP - N ) 360 , 540 , 700
360    IF ( IEV .EQ. LDEX ( KP ) ) GO TO 392
       IEV = LDEX ( KP )
392    DO 411 I = KP , N
       IEV = IEV + 1
411    LDEX ( I ) = IEV
450    KP = N
       RETURN
515    IF ( M + KP - N - ( LDEX ( KP ) + 1 ) ) 575 , 540 , 300
540    LDEX(KP)=LDEX(KP)+1
       RETURN
575    IF ( KP .EQ. 1 ) GO TO 620
       KP=KP-1
       GO TO 515
620    IND = 2
700    RETURN
       END
       FUNCTION LDET(A)
       IMPLICIT INTEGER(A-Z)
       DIMENSION A(10,10),ADEN(10,10)
       COMMON / L1 / N
       DO 122 MM = 1 , N
       DO 122 NN = 1 , N
122    ADEN ( MM , NN ) = 1
       IF(N.EQ.1)GO TO 300
       NM1=N-1
       DO 288 I=1,NM1
       IP1=I+1
       IF(A(I,I).NE.0)GO TO 201
       DO 155 J=IP1,N
       IF(A(J,I).NE.0)GO TO 175
```

```
155     CONTINUE
        LDET=0
        RETURN
175     DO 188 K=I,N
        EXTRA=A(I,K)
        A(I,K)=A(J,K)
        A(J,K)=-EXTRA
        EXTRA=ADEN(I,K)
        ADEN(I,K)=ADEN(J,K)
188     ADEN(J,K)=EXTRA
201     CALL RECIP(A(I,I),ADEN(I,I),RECNUM,RECDEN)
        DO 288 J=IP1,N
        IF(A(J,I).EQ.0)GO TO 288
        CALL MULTN(A(J,I),ADEN(J,I),RECNUM,RECDEN,FACNUM,FACDEN)
        DO 255 K=IP1,N
        IF(A(I,K).EQ.0)GO TO 255
        CALL MULTN(FACNUM,FACDEN,A(I,K),ADEN(I,K),TEMP1,TEMP2)
        CALL SUBTN(A(J,K),ADEN(J,K),TEMP1,TEMP2,TEMP3,TEMP4)
        A(J,K)=TEMP3
        ADEN(J,K)=TEMP4
255     CONTINUE
288     CONTINUE
300     NUM=1
        DEN=1
        DO 344 I=1,N
        CALL MULTN(NUM,DEN,A(I,I),ADEN(I,I),NUM1,DEN1)
        NUM = NUM1
344     DEN = DEN1
        LDET=NUM
        RETURN
        END
        SUBROUTINE SUBTN(I,J,K,L,M,N)
        IMPLICIT INTEGER(A-Z)
        DEN=J/HCF(J,L)*L
        NUM=DEN/J*I-DEN/L*K
        FAC=HCF(NUM,DEN)
        M=NUM/FAC
        N=DEN/FAC
        RETURN
        END
        SUBROUTINE MULTN(I,J,K,L,M,N)
        IMPLICIT INTEGER(A-Z)
        HCF1=HCF(I,L)
        HCF2=HCF(J,K)
        M=I/HCF1*(K/HCF2)
        N=J/HCF2*(L/HCF1)
        RETURN
        END
        SUBROUTINE RECIP(I,J,K,L)
        IMPLICIT INTEGER(A-Z)
```

```
      L=IABS(I)
      K=ISIGN(J,I)
      RETURN
      END
      INTEGER FUNCTION HCF(I,J)
      INTEGER REM
      IF(I.NE.0)GO TO 202
      HCF=IABS(J)
      RETURN
202   IF(J.NE.0)GO TO 253
      HCF=IABS(I)
      RETURN
253   KOPY1=IABS(I)
      KOPY2=IABS(J)
325   REM=MOD(KOPY1,KOPY2)
      IF(REM.NE.0)GO TO 400
      HCF=KOPY2
      RETURN
400   KOPY1=KOPY2
      KOPY2=REM
      GO TO 325
      RETURN
      END
```

//

3. ALGORITHM FOR COMPUTING THE CHARACTERISTIC ROOTS OF A MATRIX

THIS ALGORITHM IS A VERY SIMPLE ONE. ONCE WE HAVE CALCULATED THE COEFFICIENTS OF THE CHARACTERISTIC POLYNOMIAL, AND NOW THAT WE KNOW THAT IT MUST BE MONIC, WE TAKE THE HELP OF THE PROGRAM FOR CALCULATING ALL INTEGER ZEROS OF A MONIC POLYNOMIAL, WHICH WE HAVE ALREADY DEVELOPED IN CHAPTER 2. WE CHANGE THAT PROGRAM INTO A SUBROUTINE AND FINALLY WE GET THE COMPLETE PROGRAM WHICH IS GIVEN ON THE SUCCEEDING PAGES. WE MUST REMARK THAT THIS PROGRAM COMPUTES NOT ONLY ALL THE INTEGER CHARACTERISTIC ROOTS, BUT ALSO ALL THE RATIONAL CHARACTERISTIC ROOTS, BECAUSE BY THE THEORY OF EQUATIONS WE KNOW THAT IN THE CASE OF A MONIC POLYNOMIAL EVERY RATIONAL ZERO MUST BE AN INTEGER.

```
//
      DIMENSION MAT ( 10 , 10 ) , N ( 10 ) , NMAT ( 10 , 10 ) ,
     1LDEX ( 10 ) , ITZ ( 10 )
      INTEGER PR(43)
      COMMON/A1/PR/B1/N/L1/M
      CALL STORE
      WRITE ( 1 , 7 )
7     FORMAT ( 1X , 'THIS PROGRAM FINDS ALL INTEGER ZEROS OF THE'
     1,1X,'CHARACTERISTIC POLYNOMIAL' /1X ,'OF A GIVEN SQUARE MATRIX' )
79    WRITE ( 1 , 107 )
107   FORMAT ( 1X , 'TYPE THE ORDER OF THE MATRIX IN FORMAT ( I2 )'
     1/1X , 'FOR STOPPING PLEASE TYPE : 0' )
791   READ ( 1 , 127 ) M
127   FORMAT ( I2 )
      IF ( M ) 149 , 799 , 198
149   WRITE ( 1 , 167 )
167   FORMAT (1X , 'DATA ILLEGAL, KINDLY TYPE AGAIN' )
      GO TO 791
198   IF ( M .GT. 10 ) GO TO 749
      WRITE ( 1 , 227 )
227   FORMAT ( 1X ,'TYPE THE ENTRIES OF THE MATRIX IN FORMAT ( 13I6 )')
      READ ( 1 , 257 ) ( ( MAT ( I , JJ ) , JJ = 1 , M ) , I = 1 , M )
257   FORMAT ( 13I6 )
      DO 488 I = 1 , M
      N ( I ) = 0
      IF ( I .EQ. M ) GO TO 420
      IND = 0
291   CALL COMB (  I , IND , LDEX , IEV , KP  )
      GO TO ( 234 , 488 ) , IND
234   DO 411 KK = 1 , M
      DO 411 LL = 1 , M
      IF ( LL .EQ. KK ) GO TO 410
      NMAT ( LL , KK ) = 0
      GO TO 411
410   NMAT ( LL , KK ) = 1
411   CONTINUE
      DO 713 K = 1 , I
      LDEXK = LDEX ( K )
      DO 713 LL = 1 , M
713   NMAT ( LL , LDEXK ) = - MAT ( LL , LDEXK )
      N ( I ) = N ( I ) + LDET ( NMAT )
      GO TO 291
420   DO 422 J = 1 , M
      DO 422 LL = 1 , M
```

```
422   NMAT ( LL , J ) = - MAT ( LL , J )
      N ( I ) = N    I ) + LDET ( NMAT )
488   CONTINUE
      WRITE ( 1 , 3454 )
3454  FORMAT ( 1X , 'THE GIVEN MATRIX IS AS FOLLOWS' )
      DO 511 MM = 1 , M
511   WRITE ( 1 , 587 ) ( MAT ( MM , KK ) , KK = 1 , M )
587   FORMAT ( 1' , 10 ( I6 , 1X ) )
      CALL ZEROS(ITZ,NZ)
      IF(NZ.GT.0)GO TO 645
      WRITE(1,607)
607   FORMAT(1^,'THIS MATRIX HAS NO INTEGRAL CHARACTERISTIC ROOTS')
      GO TO 79
645   IF ( NZ .EQ. M ) GO TO 729
      NUMZ = M - NZ
      WRITE(1,687)NUMZ
687   FORMAT(1X,'THIS MATRIX HAS ',I2,' CHARACTERISTIC ROOTS WHICH ARE'
     1,' NOT INTEGERS')
      WRITE(1,711)NZ,(ITZ(KK),KK=1,NZ)
711   FORMAT(1X,'THE ',I2,' INTEGRAL CHARACTERISTIC ROOTS OF THIS ',
     1'MATRIX ARE'/(1X,11I7))
      GO TO 79
729   WRITE(1,737)(ITZ(KK),KK=1,M)
737   FORMAT(1X,'ALL THE CHARACTERISTIC ROOTS OF THIS MATRIX ARE',
     1' INTEGERS AND ARE'/(1X,11I7))
      GO TO 79
749   WRITE ( 1 , 787 )
787   FORMAT ( 1X , 'THE GIVEN NUMBER WILL CROSS THE DIMENSION,' , 1X
     1,'CHANGE DIMENSION' )
799   STOP
      END
       SUBROUTINE COMB (  N , IND , LDEX , IEV , HP  )
       DIMENSION LDEX(10)
       COMMON / L1 / M
       IF ( IND .NE. 0 ) GO TO 515
       IND = 1
       IEV = 0
       DO 222 I = 1 , N
222    LDEX(I)=I
       GO TO 450
300    IF ( HP - N ) 360 , 540 , 700
360    IF ( IEV .EQ. LDEX ( HP ) ) GO TO 392
       IEV = LDEX ( HP )
392    DO 411 I = HP , N
       IEV = IEV + 1
411    LDEX ( I ) = IEV
450    HP = N
       RETURN
515    IF ( M + HP - N - ( LDEX ( HP ) + 1 ) ) 575 , 540 , 300
540    LDEX(HP)=LDEX(HP)+1
       RETURN
```

```
575     IF ( KP .EQ. 1 ) GO TO 620
        KP=KP-1
        GO TO 515
620     IND = 2
700     RETURN
        END
        FUNCTION LDET(A)
        IMPLICIT INTEGER(A-Z)
        DIMENSION A(10,10),ADEN(10,10)
        COMMON / L1 / N
        DO 122 MM = 1 , N
        DO 122 NN = 1 , N
122     ADEN ( MM , NN ) = 1
        IF(N.EQ.1)GO TO 300
        NM1=N-1
        DO 288 I=1,NM1
        IP1=I+1
        IF(A(I,I).NE.0)GO TO 201
        DO 155 J=IP1,N
        IF(A(J,I).NE.0)GO TO 175
155     CONTINUE
        LDET=0
        RETURN
175     DO 188 K=I,N
        EXTRA=A(I,K)
        A(I,K)=A(J,K)
        A(J,K)=-EXTRA
        EXTRA=ADEN(I,K)
        ADEN(I,K)=ADEN(J,K)
188     ADEN(J,K)=EXTRA
201     CALL RECIP(A(I,I),ADEN(I,I),RECNUM,RECDEN)
        DO 288 J=IP1,N
        IF(A(J,I).EQ.0)GO TO 288
        CALL MULTN(A(J,I),ADEN(J,I),RECNUM,RECDEN,FACNUM,FACDEN)
        DO 255 K=IP1,N
        IF(A(I,K).EQ.0)GO TO 255
        CALL MULTN(FACNUM,FACDEN,A(I,K),ADEN(I,K),TEMP1,TEMP2)
        CALL SUBTN(A(J,K),ADEN(J,K),TEMP1,TEMP2,TEMP3,TEMP4)
        A(J,K)=TEMP3
        ADEN(J,K)=TEMP4
255     CONTINUE
288     CONTINUE
300     NUM=1
        DEN=1
        DO 344 I=1,N
        CALL MULTN(NUM,DEN,A(I,I),ADEN(I,I),NUM1,DEN1)
        NUM = NUM1
344     DEN = DEN1
        LDET=NUM
        RETURN
        END
```

```
      SUBROUTINE SUBTN(I,J,K,L,M,N)
      IMPLICIT INTEGER(A-Z)
      DEN=J/HCF(J,L)*L
      NUM=DEN/J*I-DEN/L*K
      FAC=HCF(NUM,DEN)
      M=NUM/FAC
      N=DEN/FAC
      RETURN
      END
      SUBROUTINE MULTN(I,J,K,L,M,N)
      IMPLICIT INTEGER(A-Z)
      HCF1=HCF(I,L)
      HCF2=HCF(J,K)
      M=I/HCF1*(K/HCF2)
      N=J/HCF2*(L/HCF1)
      RETURN
      END
      SUBROUTINE RECIP(I,J,K,L)
      IMPLICIT INTEGER(A-Z)
      L=IABS(I)
      K=ISIGN(J,I)
      RETURN
      END
      INTEGER FUNCTION HCF(I,J)
      INTEGER REM
      IF(I.NE.0)GO TO 202
      HCF=IABS(J)
      RETURN
202   IF(J.NE.0)GO TO 253
      HCF=IABS(I)
      RETURN
253   KOPY1=IABS(I)
      KOPY2=IABS(J)
325   REM=MOD(KOPY1,KOPY2)
      IF(REM.NE.0)GO TO 400
      HCF=KOPY2
      RETURN
400   KOPY1=KOPY2
      KOPY2=REM
      GO TO 325
      RETURN
      END
      SUBROUTINE ZEROS(ITZ,NZ)
      DIMENSION ICOF(10),ITZ(10),IFAC(96)
      COMMON/B1/ICOF/L1/ID/E1/NC,IC/C1/IFAC,NF/D1/KK
      NC=ID
      NZ=0
      DO 211 J=1,ID
      IF(ICOF(NC).NE.0)GO TO 242
      NZ=NZ+1
      ITZ(NZ)=0
```

```
 211    NC=NC-1
        RETURN
 242    IF(NC.NE.1)GO TO 275
 256    NZ=NZ+1
        ITZ(NZ)=-ICOF(1)
        RETURN
 275    NEFC=0
        KK=IABS(ICOF(NC))
        CALL FAC
 300    NEFC=NEFC+1
        IF(NEFC.GT.NF)RETURN
        IC=IFAC(NEFC)
 335    ISUM=1
        DO 377 K=1,NC
 377    ISUM=ISUM*IC+ICOF(K)
        IF(ISUM.EQ.0)GO TO 406
        IF(IC.LT.0)GO TO 300
        IC=-IC
        GO TO 335
 406    NZ=NZ+1
        ITZ(NZ)=IC
        CALL POLDIV
        IF(NC.EQ.1)GO TO 256
        GO TO 335
        END
        SUBROUTINE POLDIV
        DIMENSION ICOF(10)
        COMMON/B1/ICOF/E1/NC,ID
        NC=NC-1
        ICOF(1)=ICOF(1)+ID
        IF(NC.EQ.1)RETURN
        DO 122 I=2,NC
 122    ICOF(I)=ICOF(I)+ID*ICOF(I-1)
        RETURN
        END
        SUBROUTINE FAC
        INTEGER PRIME(43),FACTOR(96),FACT(6),POWER(6),SQRUT,DIV,QUOT,
       1PROD,FACTM,POWERM
        COMMON/A1/PRIME/C1/FACTOR,N/D1/NUMBER
        IF(NUMBER.GT.1)GO TO 7000
        N=1
        FACTOR(1)=1
        RETURN
7000    N1=NUMBER
        J=1
        NOF=0
        DIV=2
        IND=1
 45     I=1
        LIMIT=SQRUT(N1)
```

```
44      IF(DIV.GT.LIMIT)GO TO 12345
9       QUOT=N1/DIV
        IF(N1.GT.QUOT*DIV)GO TO 12350
        GO TO(6,7),I
6       NOF=NOF+1
        POWER(NOF)=1
        GO TO(12352,12353),IND
12352   FACT(NOF)=DIV
        I=2
751     N1=QUOT
        IF(N1.EQ.1)GO TO 611
        GO TO 9
7       POWER(NOF)=POWER(NOF)+1
        GO TO 751
12350   J=J+1
        DIV=PRIME(J)
        GO TO(44,45),I
12345   IND=2
        GO TO 6
12353   FACT(NOF)=N1
611     M=1
        N=1
        FACTOR(1)=1
        PROD=1
820     FACTM=FACT(M)
        POWERM=POWER(M)
        LIMIT=PROD*POWERM
        DO 92 J1=1,LIMIT
        N=N+1
92      FACTOR(N)=FACTOR(N-PROD)*FACTM
        IF(M.EQ.NOF)GO TO 100
        PROD=PROD+LIMIT
        M=M+1
        GO TO 820
100     IF(NOF.GT.1)CALL SORT
        RETURN
        END
        SUBROUTINE STORE
        INTEGER P(43),PRMSQ,PRMPRD,DIV,PMP1
        COMMON/A1/P
        P(1)=2
        P(2)=3
        P(3)=5
        P(4)=7
        P(5)=11
        P(6)=13
        K=6
        M=2
        PRMSQ=25
        PRMPRD=35
```

```
      IND=1
      N=17
539   GO TO(40,20),IND
40    IF(N.LT.PRMSQ)GO TO 100
      IND=2
      GO TO 22
20    IF(N.EQ.PRMPRD)GO TO 200
100   DO 18 J=2,M
      DIV=P(J)
      IF(N.EQ.N/DIV*DIV)GO TO 22
18    CONTINUE
      K=K+1
      P(K)=N
      IF(N.GT.181)RETURN
      GO TO 22
200   IND=1
      M=M+1
      PMP1=P(M+1)
      PRMSQ=PMP1*PMP1
      PRMPRD=PMP1*P(M+2)
22    N=N+2
      GO TO 539
      END
      INTEGER FUNCTION SQRUT(N)
      INTEGER PAIR(3),QUOT,PAIRI,SQRUT2,REM,DVND,TRDIV,TRQT
      NN=N
      I=1
57    QUOT=NN/100
      PAIR(I)=NN-100*QUOT
      IF(QUOT.EQ.0)GO TO 26
      I=I+1
      NN=QUOT
      GO TO 57
26    PAIRI=PAIR(I)
      DO 63 NLS=1,8
      SQRUT=10-NLS
      SQRUT2=SQRUT*SQRUT
      IF(PAIRI.GE.SQRUT2)GO TO 153
63    CONTINUE
      SQRUT=1
153   REM=PAIRI-SQRUT*SQRUT
1535  IF(I.EQ.1)RETURN
      I=I-1
      DVND=REM*100+PAIR(I)
      TRDIV=20*SQRUT
      TRQT=DVND/TRDIV
      IF(TRQT.GT.9)TRQT=9
9721  REM=DVND-(TRDIV+TRQT)*TRQT
      IF(REM.GE.0)GO TO 97
      TRQT=TRQT-1
```

```
      GO TO 9721
97    SQRUT=SQRUT*10+TRQT
      GO TO 153
      END
      SUBROUTINE SORT
      INTEGER FAC(96),SMALL,COPY
      COMMON/C1/FAC,L
      LM1=L-1
      DO 20 I=1,LM1
      IP1=I+1
      NSMALL=IP1
      SMALL=FAC(NSMALL)
      IF(I.EQ.LM1)GO TO 83
      IP2=I+2
      DO 16 K=IP2,L
      IF(SMALL.LE.FAC(K))GO TO 16
      NSMALL=K
      SMALL=FAC(NSMALL)
16    CONTINUE
83    IF(SMALL.GE.FAC(I))GO TO 20
      COPY=FAC(I)
      FAC(I)=SMALL
      FAC(NSMALL)=COPY
20    CONTINUE
      RETURN
      END

//
```

4. PROGRAM FOR GENERATING A SQUARE MATRIX WITH UNIT DETERMINANT

IF WE ARBITRARILY WRITE A NON-TRIVIAL SQUARE MATRIX WITH INTEGER ENTRIES, EVEN OF A VERY SMALL ORDER LIKE 3, ALMOST ALWAYS ITS DETERMINANT WILL NEVER TURN OUT TO BE +1 OR -1. SO EVEN IF THE MATRIX IS NON-SINGULAR, ITS INVERSE MATRIX WILL NEVER HAVE ALL INTEGER ENTRIES. BUT IN ORDER TO PROVIDE TESTING DATA FOR THE PREVIOUS PROGRAM, WE REQUIRE MATRICES OF THE TYPE PDP^{-1} WHERE D IS A DIAGONAL MATRIX AND BOTH P AND P^{-1} HAVE INTEGER ENTRIES. THEREFORE WE HAVE DEVELOPED THIS PROGRAM. WE GENERATE SUCH A MATRIX BY THE METHOD OF BORDERING. IT IS VERY EASY TO WRITE A SQUARE MATRIX OF ORDER 2 WITH UNIT DETERMINANT. WE FIRST WRITE SUCH A MATRIX AND THEN FILL UP THE FIRST 2 ENTRIES OF THE 3RD ROW AND THE FIRST 2 ENTRIES OF THE 3RD COLUMN ARBITRARILY. THEN BY THE HELP OF THE DETERMINANT SUBPROGRAM, WE CALCULATE P(3,3) IN SUCH A WAY THAT THE SUB-DETERMINANT FORMED BY THE FIRST 3 ROWS AND THE FIRST 3 COLUMNS WILL BE UNITY. IT IS EASY TO SEE THAT P(3,3) WILL COMPUTE TO BE AN INTEGER. THEN WE AGAIN WRITE THE FIRST 3 ENTRIES OF THE 4TH ROW AND THE FIRST 3 ENTRIES OF THE 4TH COLUMN ARBITRARILY, AND COMPUTE P(4,4) USING THE DETERMINANT SUBPROGRAM, AND SO ON. CONTINUING IN THIS WAY, WE CAN ALWAYS GENERATE A SQUARE MATRIX OF ARBITRARILY BIG ORDER SUCH THAT ITS DETERMINANT IS UNITY AND SO ITS INVERSE MATRIX WILL ALSO HAVE ALL INTEGER ENTRIES. ~~THE PROGRAM FOR INVERSION OF A MATRIX HAS ALSO BEEN WRITTEN BY MY TEACHER AND I AM REPRODUCING IT HERE FOR READY REFERENCE. I HAVE COMPUTED TEST DATA FOR MY EARLIER PROGRAM WITH THE HELP OF THIS PROGRAM.~~

```
      DIMENSION MAT(10,10),NMAT(10,10)
      WRITE(1,4000)
4000  FORMAT(' WE SHALL GENERATE MATRICES OF ORDERS < 11 IN SUCH A',
     11X,'WAY THAT IN CASE'/' THE ORDER HAPPENS TO BE GREATER THAN',
     21X,'UNITY'/' THE DETERMINANT OF THE MATRIX WILL BE UNITY')
      WRITE(1,5000)
5000  FORMAT(' BECAUSE ALL ENTRIES OF THIS MATRIX WILL BE INTEGERS',
     11X,' ALL ENTRIES OF ITS INVERSE'/' MATRIX WILL ALSO BE',
     21X,'INTEGERS')
6     WRITE(1,2000)
61    READ(1,3)K
      IF(K)15,16,17
3     FORMAT(I2)
15    WRITE(1,24)
24    FORMAT(' DATA ILLEGAL, PLEASE TYPE AGAIN')
      GO TO 61
17    IF(K.GT.1)GO TO 50
      WRITE(1,10000)
      READ(1,13)I
      MAT(1,1)=I
13    FORMAT(13I6)
      GO TO 100
50    IF(K.LE.10)GO TO 511
      WRITE(1,66)
      GO TO 61
511   WRITE(1,650)
650   FORMAT(' TYPE 4 INTEGERS I,J,L,M IN SUCH A WAY',
     11X,'THAT I*M-J*L=1')
660   READ(1,13)I,J,L,M
      IF(I*M-J*L.EQ.1)GO TO 88
      WRITE(1,666)
666   FORMAT(' SORRY, YOUR INTEGERS ARE WRONG: TYPE THEM IN SUCH',
     11X,'A WAY THAT I*M-J*L=1')
      GO TO 660
88    MAT(1,1)=I
      MAT(1,2)=J
      MAT(2,1)=L
      MAT(2,2)=M
      IF(I.EQ.2)GO TO 100
2000  FORMAT(' TYPE THE ORDER OF THE MATRIX WHICH YOU WANT',
     11X,'TO GENERATE')
10000 FORMAT(' TYPE ANY 1 INTEGER')
66    FORMAT(' DESIRED MATRIX TOO LARGE, TYPE',
     11X,'A SMALLER NUMBER AS THE ORDER OF THE MATRIX')
      ISIGN=-1
      DO 500 INDEX=3,I
      INDEX1=INDEX-1
      ISIGN=-ISIGN
      WRITE(1,77)INDEX1
```

```
 77     FORMAT(' TYPE ANY',I3,1X,'INTEGERS')
        READ(1,13)(MAT(INDEX,LL-1),LL=2,INDEX)
        WRITE(1,78)INDEX1
 78     FORMAT(' AGAIN TYPE ANY',I3,1X,'INTEGERS')
        READ(1,13)(MAT(LL-1,INDEX),LL=2,INDEX)
        NEW=1
        DO 672 JJ=1,INDEX1
        IF(JJ.GT.1)GO TO 4
        DO 270 KK=1,INDEX1
        DO 270 KKK=1,INDEX1
 270    NMAT(KK,KKK)=MAT(KK,KKK+1)
 4      NEW=NEW+(-1)**JJ*MAT(INDEX,JJ)*LDET(NMAT,INDEX1)*ISIGN
        IF(JJ.EQ.INDEX1)GO TO 672
        DO 550 LLLL=1,INDEX1
 550    NMAT(LLLL,JJ)=MAT(LLLL,JJ)
 672    CONTINUE
 500    MAT(INDEX,INDEX)=NEW
 100    DO 501 I1=1,K
 501    WRITE(1,40)(MAT(I1,J1),J1=1,K)
        NDET=LDET(MAT,K)
        WRITE(1,40)NDET
        GO TO 6
 16     STOP
 40     FORMAT(1X,10I7)
        END
         FUNCTION LDET(A,N)
         IMPLICIT INTEGER(A-Z)
         DIMENSION A(10,10),ADEN(10,10),COPYA(10,10),COPYD(10,10)
         DO 900 MM = 1 , N
         DO 900 NN = 1 , N
 900     ADEN ( MM , NN ) = 1
         DO 56 MM = 1 , N
         DO 56 NN = 1 , N
         COPYA ( MM , NN ) = A ( MM , NN )
 56      COPYD ( MM , NN ) = ADEN ( MM , NN )
         IF(N.EQ.1)GO TO 100
         NM1=N-1
         DO 200 I=1,NM1
         IP1=I+1
         IF(COPYA(I,I).NE.0)GO TO 23
         DO 20 J=IP1,N
         IF(COPYA(J,I).NE.0)GO TO 75
 20      CONTINUE
         LDET=0
         RETURN
 75      DO 50 K=I,N
         EXTRA = COPYA ( I , K )
         COPYA(I,K)=COPYA(J,K)
         COPYA(J,K)=-EXTRA
         EXTRA=COPYD(I,K)
```

```
      COPYD(I,K)=COPYD(J,K)
50    COPYD(J,K)=EXTRA
23    CALL RECIP(COPYA(I,I),COPYD(I,I),RECNUM,RECDEN)
      DO 200 J=IP1,N
      IF(COPYA(J,I).EQ.0)GO TO 200
      CALL MULTN(COPYA(J,I),COPYD(J,I),RECNUM,RECDEN,FACNUM,FACDEN)
      DO 700 K=IP1,N
      IF(COPYA(I,K).EQ.0)GO TO 700
      CALL MULTN(FACNUM,FACDEN,COPYA(I,K),COPYD(I,K),TEMP1,TEMP2)
      CALL SUBTN(COPYA(J,K),COPYD(J,K),TEMP1,TEMP2,TEMP3,TEMP4)
      COPYA(J,K)=TEMP3
      COPYD(J,K)=TEMP4
700   CONTINUE
200   CONTINUE
100   NUM=1
      DEN=1
      DO 500 I=1,N
      CALL MULTN(NUM,DEN,COPYA(I,I),COPYD(I,I),NUM1,DEN1)
      NUM = NUM1
500   DEN = DEN1
      IF(DEN.NE.1)GO TO 777
      LDET=NUM
      RETURN
777   WRITE(1,778)
778   FORMAT(' THERE IS SOME MISTAKE IN THE CALCULATIONS')
      STOP
      END
      SUBROUTINE SUBTN(I,J,K,L,M,N)
      IMPLICIT INTEGER(A-Z)
5     DEN=J/HCF(J,L)*L
8     NUM=DEN/J*I-DEN/L*K
1     FAC=HCF(NUM,DEN)
2     M=NUM/FAC
3     N=DEN/FAC
      RETURN
      END
       SUBROUTINE MULTN(I,J,K,L,M,N)
      IMPLICIT INTEGER(A-Z)
      HCF1=HCF(I,L)
      HCF2=HCF(J,K)
15    M=I/HCF1*(K/HCF2)
16    N=J/HCF2*(L/HCF1)
      RETURN
      END
      SUBROUTINE RECIP(I,J,K,L)
      IMPLICIT INTEGER(A-Z)
      L=IABS(I)
      K=ISIGN(J,I)
      RETURN
      END
```

```
      INTEGER FUNCTION HCF(I,J)
      INTEGER REM
      IF(I.NE.0)GO TO 85
      HCF=IABS(J)
      RETURN
85    IF(J.NE.0)GO TO 153
      HCF=IABS(I)
      RETURN
153   KOPY1=IABS(I)
      KOPY2=IABS(J)
900   REM=MOD(KOPY1,KOPY2)
      IF(REM.NE.0)GO TO 815
      HCF=KOPY2
      RETURN
815   KOPY1=KOPY2
      KOPY2=REM
      GO TO 900
      RETURN
      END

//
```

BIBLIOGRAPHY

1. Computer Programming In Fortran IV - Dr. Vijai Kumar (Vol. I)

2. Computer Programming In Fortran IV - Dr. Vijai Kumar (Vol. II, in press)

3. Numerical algorithms: Computations in Science and Engineering, Affiliated East West Press, New Delhi, pp.107-110.

4. Programming with Fortran 77 - Ram Kumar

5. Fortran IV Computer programming - K.D. Sharma

6. Fortran and Other Languages - PVS Rao

7. Proceedings workshop on Computer Applications in Continuum Mechanics (March 11-13, 1986), Deptt. of Mathematics, University of Roorkee, Roorkee, India.

8. The Art of Computer Programming (Vol.I) Addision, Addison-Wesley, M assa chussetts. —KNUTH, D.E. (1968)