Computer Sciences Department
The University of Wisconsin
1210 West Dayton Street
Madison, Wisconsin 53706

# THE SAC-1 POLYNOMIAL GCD AND RESULTANT SYSTEM*

by

George E. Collins

Technical Report #145†

February 1972

*Research supported by NSF grants GJ-239 and GJ-30125X, and by the University of Wisconsin - Madison Academic Computing Center and Graduate School Research Committee.

† This report also appears as University of Wisconsin Madison Academic Computing Center Technical Report No. 27 .

May 30, 1972

# ERRATUM

## The SAC-1 Polynomial GCD and Resultant System

Page 61. In the listing for CPDIV insert just preceding the END statement, the statement "110 GO TO (2,71),RR".

George E. Collins

# ERRATA

## The SAC-1 Polynomial GCD and Resultant System

Page 2, lines 5-7. The subprogram PPOWER from the Polynomial Real Zero System is also required.

Page 26. Change steps (10) and (12) of CPQDB to the following:

(10) If $\overline{A}\neq 0$, (if G<0, go to (12); return); if FIRST($\overline{A}$)$\neq$0, go to (11); DECAP(W,$\overline{A}$); go to (10).

(12) If C$\neq$0, (T$\leftarrow$TYPE(C); DECAP(G,C); if T=1, erase G; go to (12)); C$\leftarrow$-1; return.

Page 67, lines 3-13. Change to the following:

```
100  IF (ABAR.NE.0) GO TO 101
     IF (G.LT.0) GO TO 120
     GO TO 130
101  IF (FIRST(ABAR).NE.0) GO TO 110
     CALL DECAP(W,ABAR)
     GO TO 100
110  CALL DECAP(W,ABAR)
     CALL CPERAS(W)
     IF (ABAR.NE.0) GO TO 110
120  IF (C.EQ.0) GO TO 121
     T=TYPE(C)
     CALL DECAP(G,C)
     IF (T.EQ.1) CALL CPERAS(G)
     GO TO 120
121  C=-1
130  GO TO (2,61),RR
     END
```

Page 31, line 9. After "B$\leftarrow$PFA(D,B)" insert "; go to (4)".

Page 39, line -3. Change "CRECIP(p,B)" to "CRECIP(p,b)".

Page 61, line -18. Preceding "END" insert "110 GO TO (2,71),RR".

Page 78, line -23. Change "GO TO 150" to "GO TO 50".

Page 78, line -18. Following "CALL ERLA(L)" insert "L=LST".

Page 90, line -2. Change "*5X**1" to "-5X**1".

Page 94, line 8. Change "CPDIV....24" to CPTDIV...24".

Page 32, line 5. Add "The input polynomial C, when r>1, is erased.".

I am grateful to M. Rothstein and O. Leringe for the above corrections.

July 8, 1974 George E. Collins

ERRATUM

The SAC-1 Polynomial GCD and Resultant System (C.S. Tech. Report No. 145, MACC Report No. 27)

Page 34, line 7. Change "B ← TAIL (FIRST (B))." to "B ← FIRST (TAIL(B))."

George E. Collins
September 21, 1972

June 14, 1972

# ERRATA

## The SAC-1 Polynomial GCD and Resultant System

(1) CPUCNT

Page 13. In the first line of the specifications, delete the word "non-zero". In the last line of the specifications, add the sentence "If A = 0, then B = 0.".

Page 14. In step (2) of the algorithm, insert "If C = 0, go to (4);" preceding "L ← TAIL(C)".

Page 70. In the program listing, replace "20 L = TAIL(C)" by:

```
20  IF (C.EQ.0) GO TO 40
    L = TAIL(C)
```

(2) CPUDIV

Page 28. In the first line of the specifications, delete the word "non-zero". In step (1) of the algorithm, insert "If A = 0, (C ← 0; return);" preceding "If TYPE(TAIL(A)) = 0,".

Page 71. In the program listing, replace

```
10  IF (TYPE(TAIL(A)).NE.0) GO TO 11
    C = CPQ(P,A,B)
    GO TO 40
11  N = FIRST(A)
```

by

```
10  IF (A.NE.0) GO TO 11
    C = 0
    GO TO 40
11  IF (TYPE(TAIL(A)).NE.0) GO TO 12
    C = CPQ(P,A,B)
    GO TO 40
12  N = FIRST(A)
```

Errata Continued.

(3) CPUPR

Page 27. In the first line of the specifications, delete the word "non-zero". In step (1) of the algorithm insert "If A = 0, (C ← 0; return);" preceding "If TYPE(TAIL(A)) = 0,".

Page 72. In the program listing, replace

```
10  IF (TYPE(TAIL(A)).NE.0) GO TO 11
    C = CPPROD(P,A,B)
    GO TO 40
11  N = FIRST(A)
```

by

```
10  IF (A.NE.0) GO TO 11
    C = 0
    GO TO 40
11  IF (TYPE(TAIL(A)).NE.0) GO TO 12
    C = CPPROD(P,A,B)
    GO TO 40
12  N = FIRST(A)
```

(4) INDEX OF ALGORITHMS

Page 94. Change "CPUNCT" to "CPUCNT".
Change "CPURP" to "CPUPR".

George E. Collins

# THE SAC-1 POLYNOMIAL GCD AND RESULTANT SYSTEM

by

George E. Collins

## ABSTRACT

This is the eighth in the series of SAC-1 subsystems for Symbolic and Algebraic Calculation. The present subsystem provides programs for computing the greatest common divisors and resultants of multivariate polynomials, which are based on the new and much faster modular algorithms of W. S. Brown and G. E. Collins. The system also contains modular-algorithm programs for polynomial multiplication and trial division, and improved programs for the Chinese remainder theorem and interpolation. This report contains, for each program in the system, a user's functional specification, a formal algorithm description, a theoretical computing time, and a Fortran program listing. Illustrative empirical computing times are given for many of the programs, and a test program is included for assistance in implementation and use of the system.

TABLE OF CONTENTS

## 1. Introduction

The SAC-1 Polynomial GCD and Resultant System is the eighth subsystem of the SAC-1 System for Symbolic and Algebraic Calculation. Previous subsystems are the List Processing System [5], the Integer Arithmetic System [6], the Polynomial System [7], the Rational Function System [8], the Modular Arithmetic System [9], the Rational Function Integration System [10], and the Polynomial Real Zero System [11].

The List Processing and Integer Arithmetic subsystems provide fundamental capabilities required as a foundation for all subsequent subsystems, each of which provides some additional capability for performing various operations on polynomials or rational functions with exact numerical coefficients.

The primary new capabilities supplied by the present subsystem are for the calculation of the greatest common divisor (g.c.d.) and resultant of multivariate polynomials with (exact) infinite-precision integer coefficients. A polynomial g.c.d. capability was previously provided by the Polynomial System, but the present system contains a new modular algorithm for polynomial g.c.d. calculation which is much faster. The algorithm used for resultant calculation in the present system is also a modular algorithm, which is much faster than any previously considered algorithms. For a thorough discussion

of the theory supporting these algorithms and for theoretical analyses of their computing times, see [2], [3] and [4].

Implementation of the Polynomial GCD and Resultant System requires prior implementation of the List Processing, Integer Arithmetic, Polynomial, and Modular Arithmetic systems; in addition two subprograms, ELPOF2 and PNORMF, are required from the Polynomial Real Zero system. When the Polynomial GCD and Resultant System is implemented, the two subprograms PGCD and PCONT must be deleted from the Polynomial System; the Polynomial GCD and Resultant System contains new programs with the same names and functions (computing polynomial g.c.d.'s and contents, respectively) but using improved algorithms. The subprograms PCGCD and PCGCDA may then also be deleted from the Polynomial system since they will be superfluous for most purposes, but no harm will result from retaining them. The subprograms PPP (polynomial primitive part) and PCPP (polynomial content and primitive part) in the Polynomial System should be retained; they will automatically use the new modular g.c.d. algorithm.

The Polynomial GCD and Resultant System also contains new modular algorithms for polynomial multiplication and division. While these new algorithms generally have theoretical computing times which are superior to those of the classical algorithms of the

Polynomial System, their empirically observed computing times tend to be greater than those of the classical algorithms when the input and output polynomials are relatively small in coefficient size or degree. For larger polynomials, the modular multiplication and division algorithms are indeed observed to be faster. Therefore these new algorithms have been given distinct names so that the classical algorithms can be retained, and the user can exercise judgement as to which algorithm to apply in a given case. Unfortunately, it appears very difficult to devise a satisfactory algorithm to decide automatically whether the classical or modular algorithm would be most efficient for given inputs. Thus, for example, the present system provides the modular algorithm PMPY for polynomial multiplication as an alternative to the classical algorithm PPROD of the polynomial System; and PTDIV is provided as an alternative to PQ for polynomial trial division. The use of PDIV is appropriate when the quotient is known to exist and the inputs are large.

The Polynomial GCD and Resultant System also provides new algorithms for the Chinese remainder theorem and interpolation. The new algorithms are based on the same mathematical formulas as the corresponding algorithms of the Modular Arithmetic System, but care has been taken to eliminate certain extraneous or redundant calculations, with a resulting significant improvement in performance. In this case, again, new names have been used for the new algorithms,

but for a different reason, namely that the new algorithms have different inputs or outputs.

The present system contains other new programs, too numerous and varied to detail, and some are primarily auxilliary to the main programs. In this respect, the present system may be regarded as supplemental to the Polynomial and Modular Arithmetic Systems.

This SAC-1 report continues our previously established practice of appending to each algorithm description a statement of its theoretical computing time. The present report, however, now uses the notions and notations of dominance and codominance in place of O-notation. For an introduction to these concepts and their application, see [4] or [12]. Thus, Section 2 contains for each program of the system (a) a user's functional specification, (b) an algorithm description, and (c) a theoretical computing time, in this order. For purposes of convenient reference, the algorithms are arranged by category and an algorithm index is provided.

The theoretical computing times of Section 2 have the important advantage of being computer-independent. For this very reason they can not, by themselves, be used to predict the computing time of a given program for specified inputs on a particular computer. Therefore, in Section 3 empirically observed computing times are presented for a particular computer, the UNIVAC 1108, for a variety

of inputs to the major programs of the system (i.e. those most likely to be applied directly by the user, and those of most interest). Combination of the theoretical and empirical computing times will usually yield a reasonably accurate estimate of the running time of a problem, or permit an assessment of the feasibility of a proposed application.

Section 4 contains a listing of each program of the system. These are in alphabetical order by name and are provided so that the programs can be keypunched for compilation.

Section 5 contains a main program which can be used to test implementation of the system. This is by no means an exhaustive test program, but it will often assist in detecting many errors. This program also serves to illustrate some features of any main program written to apply the system.

The programs of this system are all written in Fortran IV, in strict conformity to the USASI specifications, [1]. The system has been thoroughly tested, but errors may still persist, and the author would appreciate notification of any which are discovered. This manual is intended to be relatively self-contained, but a degree of familiarity with previous SAC-1 systems, especially [5], [6], [7] and [9], is assumed.

## 2. Algorithm Descriptions

The algorithms of this system are classified below into eight categories. Within each category, those algorithms most likely to be applied directly by a user are presented first. For each algorithm there is given first the functional specifications, then an algorithm description, and finally a statement about the theoretical computing time of the algorithm.

In the functional specifications and computing time statements, the following conventions and notations are used. When a polynomial $A$ is denoted by $A(x_1,\ldots,x_r)$, $x_r$ is always the main or outer variable and $x_1$ is always the inner variable. $A$ is then represented as a polynomial in $x_r$ whose coefficients are polynomials in $x_1,\ldots,x_{r-1}$. The degree of $A$ in $x_i$ is denoted by $\partial_i(A)$. When an operation is to be performed on two or more polynomials, these polynomials are assumed, except as otherwise indicated, to be compatible in the sense that they contain the same variables and these variables appear in the same order in each. For polynomials in any number of variables with integer coefficients, two norms are defined. $|A|_1$ is the sum of the absolute values of the integer coefficients of $A$ and $|A|_\infty$ is the maximum of the absolute values of the integer coefficients of $A$. These norms have the following important properties:

$$|A|_\infty \leq |A|_1 ,$$

$$|A + B|_\infty \leq |A|_\infty + |B|_\infty ,$$

$$|A + B|_1 \leq |A|_1 + |B|_1 ,$$

$$|AB|_\infty \leq |A|_\infty \quad |B|_1 ,$$

$$|AB|_1 \leq |A|_1 \quad |B|_1 .$$

For any integer $a$, we define $L_\beta(a) = 1$ if $a = 0$ and $L_\beta(a) = \lceil \log_\beta(|a| + 1) \rceil$ for $a \neq 0$. $L_\beta(a)$ is the number of digits in the base $\beta$ radix representation of $a$, if $a \neq 0$. In most contexts the base $\beta$ is fixed. Furthermore we have $L_\beta(a) \sim L_\gamma(a)$ for any two bases $\beta$ and $\gamma$. Hence we generally omit the subscript $\beta$ and write simply $L(a)$, referring to $L(a)$ as the length of $a$. This notation, which is used in many of the following computing time statements, is due to D. R. Musser [12].

## Polynomial Greatest Common Divisors

The following modular algorithms for polynomial greatest common divisor calculation are virtually identical to those described by Brown in [2], who developed and analyzed these algorithms simultaneously with myself. However, the versions given here employ the method of cofactor calculation as opposed to trial division, and the credit for this important innovation belongs entirely to Brown.

PGCDCF($A_1'$,$A_2''$,$B'$,$C_1''$,$C_2''$)

The inputs $A_1'$ and $A_2''$ are non-zero integral polynomials in r variables, $r \geq 0$. $B''$, $C_1''$ and $C_2''$ are outputs. $B''$ is the greatest common divisor of $A_1'$ and $A_2'$, a polynomial whose leading numerical coefficient is positive. $C_1'' = A_1''/B'$ and $C_2' = A_2''/B'$.

Algorithm

(1) If TYPE($A_1'$) $\neq 0$, go to (2); $B' \leftarrow$ IGCD($A_1'$,$A_2''$); $C_1'' \leftarrow$ IQ($A_1''$,$B'$); $C_2' \leftarrow$ IQ($A_2''$,$B'$); return.

(2) $a_1' \leftarrow$ PICONT($A_1''$); $A_1 \leftarrow$ PID($A_1''$,$a_1''$); $a_2'' \leftarrow$ PICONT($A_2'$); $A_2 \leftarrow$ PID($A_2'$,$a_2''$); $b' \leftarrow$ IGCD($a_1'$,$a_2'$).

(3) $a_1 \leftarrow$ PLNCF($A_1$); $a_2 \leftarrow$ PLNCF($A_2$); $\bar{b} \leftarrow$ IGCD($a_1$,$a_2$); erase $a_1$, $a_2$.

(4) $\ell_1 \leftarrow$ PNORMF($A_1$); $\ell_2 \leftarrow$ PNORMF($A_2$); $S \leftarrow$ ICOMP($\ell_1$,$\ell_2$); if $S > 0$, ($\ell \leftarrow \ell_1$; erase $\ell_2$); if $S \leq 0$, ($\ell \leftarrow \ell_2$; erase $\ell_1$); $t \leftarrow$ BORROW($\bar{b}$); IMFI(t,2); $h \leftarrow$ IPROD($\ell$,t); erase $\ell$,t; $P \leftarrow$ PRIME; $L \leftarrow 0$; $V \leftarrow$ INV(PVLIST($A_1$)).

(5) If $P = 0$, stop; ADV(p,P); $\bar{b}^* \leftarrow$ CMOD(p,$\bar{b}$); if $\bar{b}^* = 0$, go to (5).

(6) $A_1^* \leftarrow$ CPMOD(p,$A_1$); $A_2^* \leftarrow$ CPMOD(p,$A_2$).

(7) CGCDCF(p,$A_1^*$,$A_2^*$,$B^*$,$\bar{C}_1^*$,$\bar{C}_2^*$).

(8) Erase $A_1^*$,$A_2^*$; if CPONE($B^*$) $\neq 1$, go to (10).

(9) If $L \neq 0$, erase L,$\bar{B}$,$\bar{C}_1$,$\bar{C}_2$,Q; erase $B^*$,$\bar{C}_1^*$,$\bar{C}_2^*$,$\bar{b}$,h; $V \leftarrow$ INV(V); $t \leftarrow$ PFA(1,0); $B \leftarrow$ PORDER(t,V); erase t, V; $C_1 \leftarrow A_1$; $C_2 \leftarrow A_2$; go to (17).

(10) $\bar{B}^* \leftarrow$ CSPROD(p,$B^*$,$\bar{b}^*$,0); erase $B^*$.

(11) $L^* \leftarrow$ CPDEGS($\bar{B}^*$); if $L = 0$, ($L \leftarrow L^*$; go to (12)); $S \leftarrow$ ICOMP(L,$L^*$); if $S = 0$, (erase $L^*$; go to (13)); if $S < 0$, (erase $\bar{B}^*$,$\bar{C}_1^*$,$\bar{C}_2^*$,$L^*$; go to (5)); if $S > 0$, (erase $\bar{B}$,$\bar{C}_1$,$\bar{C}_2$,Q,L; $L \leftarrow L^*$).

(12) $\bar{B} \leftarrow 0$; $\bar{C}_1 \leftarrow 0$; $\bar{C}_2 \leftarrow 0$; $Q \leftarrow$ PTA(1,0).

(13) $q \leftarrow$ CRECIP(p,CMOD(p,Q)); $T \leftarrow$ CPCRA($Q,\bar{B},p,\bar{B}^*,q,V$); erase $\bar{B},\bar{B}^*$; $\bar{B} \leftarrow T$; $T \leftarrow$ CPCRA($Q,\bar{C}_1,p,\bar{C}_1^*,q,V$); erase $\bar{C}_1,\bar{C}_1^*$; $\bar{C}_1 \leftarrow T$; $T \leftarrow$ CPCRA ($Q,\bar{C}_2,p,\bar{C}_2^*,q,V$); erase $\bar{C}_2,\bar{C}_2^*$; $\bar{C}_2 \leftarrow T$; IMFI(Q,p).

(14) $S \leftarrow$ ICOMP(Q,h); if $S \leq 0$, go to (5); $m \leftarrow$ PNORMF($\bar{B}$); IMFI(m,2); $n_1 \leftarrow$ PNORMF($\bar{C}_1$); $n_2 \leftarrow$ PNORMF($\bar{C}_2$); $m_1 \leftarrow$ IPROD(m,$n_1$); $m_2 \leftarrow$ IPROD(m,$n_2$); $t_1 \leftarrow$ ICOMP(Q,$m_1$); $t_2 \leftarrow$ ICOMP(Q,$m_2$); erase $m,n_1,n_2,m_1,m_2$; if $t_1 \leq 0$ or $t_2 \leq 0$, go to (5).

(15) Erase $A_1,A_2,\bar{b},Q,L,V,h$.

(16) $d \leftarrow$ PICONT($\bar{B}$); $B \leftarrow$ PID($\bar{B}$,d); erase $\bar{B}$,d; $b \leftarrow$ PLNCF(B); $C_1 \leftarrow$ PID($\bar{C}_1$,b); erase $\bar{C}_1$;$C_2 \leftarrow$ PID($\bar{C}_2$,b); erase $\bar{C}_2$,b.

(17) $B' \leftarrow$ PIP(B,b'); erase B; $d \leftarrow$ IQ($a_1'$,b'); erase $a_1'$; $C_1' \leftarrow$ PIP($C_1$,d); erase $C_1$,d; $d \leftarrow$ IQ($a_2'$,b'); erase $a_2'$,b'; $C_2' \leftarrow$ PIP($C_2$,d); erase $C_2$,d; return.

Computing Time: It is conjectured that the average computing time is $\leq \alpha^2\lambda^r + \alpha\lambda^{r+1}$, where $\lambda = \max_{1\leq i\leq r}\{\max(\partial_i(A_1'), \partial_i(A_2'))\} + 1$ and $\alpha = \max(L(|A_1'|_1), L(|A_2'|_1))$.

C = PGCD(A,B)

A and B are integral polynomials in r variables, $r \geq 0$. C is the greatest common divisor of A and B, a positive or zero polynomial.

Algorithm

(1) If A = 0, (C ← PABS(B); return); if B = 0, (C ← PABS(A); return).

(2) If TYPE(A) = 0, (C ← IGCD(A,B); return).

(3) PGCDCF(A,B,C,$\bar{A}$,$\bar{B}$); erase $\bar{A}$, $\bar{B}$; return.

Computing Time: It is conjectured that the average computing time is $\leq \alpha^2\lambda^r + \alpha\lambda^{r+1}$, where $\lambda = \max_{1\leq i\leq r}(\max(\partial_i(A), \partial_i(B))) + 1$ and $\alpha = \max(L(|A|_1), L(|B|_1))$.

B = PCONT(A)

A is a polynomial with integer coefficients in r variables, $r \geq 1$. B is the content of A, a polynomial in r - 1 variables. If A = 0, then B = 0. Otherwise if $A(x_1,\ldots,x_r) = \sum_{i=0}^{n} A_i(x_1,\ldots,x_{r-1}) \cdot x_r^i$, then B is the g.c.d. of the coefficients $A_i$. If $B \neq 0$, then B is a positive polynomial.

Algorithm

(1) $B \leftarrow 0$; if $A = 0$, return; $T \leftarrow$ TAIL(A).

(2) ADV(C,T); $D \leftarrow$ PGCD(B,C); erase B; $B \leftarrow D$; if PONE(B) = 1, return; $T \leftarrow$ TAIL(T); if $T \neq 0$, go to (2); return.

Computing Time: It is conjectured that the average computing time is $\leq \alpha^2\lambda^r + \alpha\lambda^{r+1}$, where $\lambda = \max_{1\leq i\leq r} \partial_i(A) + 1$ and $\alpha = L(|A|_1)$.

I = PICONT(A)

A is a non-zero polynomial with integer coefficients in r variables, $r \geq 1$. I is the greatest common divisor of the integer coefficients of A, a positive infinite-precision integer.

Algorithm

(1) $B \leftarrow A$; $I \leftarrow 0$; $S \leftarrow 0$.

(2) $B \leftarrow$ TAIL(B); $C \leftarrow$ FIRST(B); if TYPE(C) = 0, go to (3); $S \leftarrow$ PFA(TAIL(TAIL(B)), S); $B \leftarrow C$; go to (2).

(3) $T \leftarrow$ IGCD(I,C); ERLA(I); $I \leftarrow T$; if FIRST(I) = 1 and TAIL(I) = 0, (ERLA(S); return); $B \leftarrow$ TAIL(TAIL(B)); if B = 0, go to (4); $C \leftarrow$ FIRST(B); go to (3).

(4) If S = 0, return; DECAP(B,S); if B = 0, go to (4); $T \leftarrow$ FIRST(B); $S \leftarrow$ PFA(TAIL(TAIL(B)),S); $B \leftarrow T$; go to (2).

Computing Time: $\leq L(a)^2 \prod_{i=1}^{r} \mu_i$, where $a = |A|_\infty$, $m_i = \partial_i(A)$ and $\mu_i = m_i + 1$.

CGCDCF(p,$A'_1$,$A'_2$,B',$C'_1$,$C'_2$)

p, $A'_1$ and $A'_2$ are inputs. $A'_1$ and $A'_2$ are non-zero polynomials over GF(p) in r variables, $r \geq 1$. B', $C'_1$ and $C'_2$ are outputs. B' is the greatest common divisor of $A'_1$ and $A'_2$, a polynomial whose leading numerical coefficient is 1. $C'_1 = A'_1/B'$ and $C'_2 = A'_2/B'$.

Algorithm

(1) $r \leftarrow$ CPNV($A'_1$); if $r > 1$, go to (3).

(2) $B' \leftarrow$ CPGCD1(p,$A'_1$,$A'_2$); $C'_1 \leftarrow$ CPQ(p,$A'_1$,B'); $C'_2 \leftarrow$ CPQ(p,$A'_2$,B'); return.

(3) CPUCPP(p,$A'_1$,$a'_1$,$A_1$); CPUCPP(p,$A'_2$,$a'_2$,$A_2$); $b' \leftarrow$ CPGCD1(p,$a'_1$,$a'_2$).

(4) $a_1 \leftarrow$ CPLUCF($A_1$); $a_2 \leftarrow$ CPLUCF($A_2$); $\bar{b} \leftarrow$ CPGCD1(p,$a_1$,$a_2$); erase $a_1$, $a_2$.

(5) $h \leftarrow$ CPDEG($\bar{b}$); $\ell_1 \leftarrow$ CPDEG1($A_1$) + h. $\ell_2 \leftarrow$ CPDEG1($A_2$) + h; $\ell \leftarrow$ max ($\ell_1$,$\ell_2$); $k \leftarrow 0$; $e \leftarrow -1$; $L \leftarrow 0$.

(6) $e \leftarrow e + 1$; if $e = p$, stop; $\bar{b}^* \leftarrow$ CPEVAL(p,$\bar{b}$,e); if $\bar{b}^* = 0$, go to (6).

(7) $A_1^* \leftarrow$ CPUEV(p,$A_1$,e); $A_2^* \leftarrow$ CPUEV(p,$A_2$,e).

(8) CGCDCF(p,$A_1^*$,$A_2^*$,$B^*$,$\bar{C}_1^*$,$\bar{C}_2^*$).

(9) Erase $A_1^*$, $A_2^*$; if CPONE($B^*$) $\neq 1$, ($\bar{B}^* \leftarrow$ CSPROD(p,$B^*$, $\bar{b}^*$, 0); erase $B^*$; go to (10)); erase $\bar{b}$, $\bar{C}_1^*$, $\bar{C}_2^*$, L; if $k \neq 0$, erase $\bar{B}$, $\bar{C}_1$, $\bar{C}_2$, D; $B \leftarrow$ PFA(0,PFL($B^*$,0)); $C_1 \leftarrow A_1$; $C_2 \leftarrow A_2$; go to (16).

(10) $L^* \leftarrow$ CPDEGS($\bar{B}^*$); if $L = 0$, ($L \leftarrow L^*$; go to (11)); $S \leftarrow$ ICOMP(L,$L^*$); if $S = 0$, (erase $L^*$; if $k = 0$, go to (11); go to (12)); if $S < 0$, (erase $\bar{B}^*$, $\bar{C}_1^*$, $\bar{C}_2^*$, $L^*$; go to (6)); if $S > 0$, (erase $\bar{B}$, $\bar{C}_1$, $\bar{C}_2$, D, L; $L \leftarrow L^*$).

(11) $\bar{B} \leftarrow 0$; $\bar{C}_1 \leftarrow 0$; $\bar{C}_2 \leftarrow 0$; $D \leftarrow$ PFA(0,PFA(1,0)); $k \leftarrow 0$; $m \leftarrow 0$; $n_1 \leftarrow 0$; $n_2 \leftarrow 0$.

(12) $E \leftarrow$ BORROW(D); CPINTI(p,$\bar{B}$,e,$\bar{B}^*$,E,r,t); if $t = 1$, $m \leftarrow k$; erase E; $E \leftarrow$ BORROW(D); CPINTI(p,$\bar{C}_1$,e,$\bar{C}_1^*$,E,r,t); if $t = 1$, $n_1 \leftarrow k$; erase E; CPINTI(p,$\bar{C}_2$,e,$\bar{C}_2^*$,D,r,t); if $t = 1$, $n_2 \leftarrow k$; $k \leftarrow k + 1$.

(13) If $m + n_1 > \ell_1$ or $m + n_2 > \ell_2$, (erase $\bar{B}, \bar{C}_1, \bar{C}_2$, D; $k \leftarrow 0$; go to (6)); if $k \leq \ell$, go to (6).

(14) Erase $A_1$, $A_2$, $\bar{b}$, D, L.

(15) CPUCPP(p,$\bar{B}$,d,B); erase $\bar{B}$,d; $b \leftarrow$ CPLUCF(B); $C_1 \leftarrow$ CPUDIV(p,$\bar{C}_1$,b); erase $\bar{C}_1$; $C_2 \leftarrow$ CPUDIV(p,$\bar{C}_2$,b); erase $\bar{C}_2$,b.

(16) $B' \leftarrow$ CPUPR(p,B,b'); erase B; $d \leftarrow$ CPQ(p,$a'_1$,b'); erase $a'_1$; $C'_1 \leftarrow$ CPUPR(p,$C_1$,d); erase $C_1$, d; $d \leftarrow$ CPQ(p,$a'_2$,b'); erase $a'_2$, b'; $C'_2 \leftarrow$ CPUPR(p,$C_2$,d); erase $C_2$, d; return.

Computing Time: It is conjectured that the average computing time is $\leq \lambda^{r+1}$, where $\lambda = \max_{1 \leq i \leq r} \max(\partial_i(A'_1), \partial_i(A'_2)) + 1$.

B = CPUCNT(p,A)

A is a non-zero polynomial over GF(p) in r variables, $r \geq 2$. B is the univariate content of A, that is, if $A(x_1,\ldots,x_r)$ is regarded as a polynomial in $x_2,\ldots,x_r$ with coefficients from GF(p)$[x_1]$, then $B = B(x_1)$ is the greatest common divisor of these coefficients, a monic polynomial.

Algorithm

(1) $B \leftarrow 0$; $C \leftarrow A$; $S \leftarrow 0$.

(2) L ← TAIL(C); if TYPE(L) = 0, go to (3); S ← PFA(TAIL(L), S); C ← FIRST(L); go to (2).

(3) If B = 0, (B ← BORROW(C); go to (4)); D ← CPGCD1(p,B,C); erase B; B ← D; if CPONE(B) = 1, (erase S; return).

(4) If S = 0, return; DECAP(L,S); if L = 0, go to (4); ADV(C,L); S ← PFA(L,S); go to (2).

Computing Time: $\leq \nu_1^2 \Pi_{i=2}^{r} \nu_i$, where $n_i = \partial_i(A)$ and $\nu_i = n_i + 1$.

CPUCPP(p,A,c,$\bar{A}$)

A is a non-zero polynomial over GF(p) in r variables, $r \geq 2$, where p and A are inputs. The output c is the univariate content of A, and the output $\bar{A}$ is the univariate primitive part of A, defined by $A = c \cdot \bar{A}$.

Algorithm

(1) c ← CPUCNT(p,A); t ← CPONE(c); if t = 1, $\bar{A}$ ← BORROW(A); if t = 0, $\bar{A}$ ← CPUDIV(p,A,c); return.

Computing Time: $\leq \nu_1^2 \Pi_{i=2}^{r} \nu_i$, where $n_i = \partial_i(A)$ and $\nu_i = n_i + 1$.

## Polynomial Resultants

The following modular resultant algorithms are as described in [4], except that evaluation and interpolation are done with respect to $x_1$ rather than $x_{r-1}$. This change results in some increase in speed for $r \geq 3$.

C = PRES(A,B)

A and B are compatible polynomials with infinite-precision integer coefficients in r variables, $r \geq 1$, with $\deg(A) \geq \deg(B) > 0$. C is the resultant of A and B, with respect to the main variable of A and B, a polynomial with integer coefficients in r - 1 variables (an integer if r = 1).

Algorithm

(1) d ← PMNORM(A); e ← PMNORM(B); m ← PDEG(A); n ← PDEG(B); f ← $2d^n e^m (m+n)!$; erase d and e.

(2) I ← PRIME; Q ← PFA(1,0); C ← 0; L ← INV(PVLIST(FIRST(TAIL(A)))).

(3) If I = 0, stop; ADV(p,I).

(4) $A^*$ ← CPMOD(p,A); if CPDEG($A^*$) < m, erase $A^*$ and go to (3).

(5) $B^*$ ← CPMOD(p,B); if CPDEG($B^*$) < n, erase $A^*$ and $B^*$ and go to (3).

(6) $C^*$ ← CPRES(p,$A^*$,$B^*$); erase $A^*$ and $B^*$.

(7) q ← CRECIP(p,CMOD(p,Q)); T ← CPCRA(Q,C,p,$C^*$,q,L); erase C; C ← T; if L ≠ 0, erase $C^*$.

(8) Q ← Q · p; S ← ICOMP(Q,f); if S < 0, go to (3).

(9) Erase f, L and Q; return.

Computing Time: $\leq K_r \sum_{i=1}^{r-1} \{S_{i,r} \prod_{j=i}^{r-1} C_{j,r}\} + K_r \{\prod_{j=1}^{r-1} C_{j,r}\} \{\sum_{j=1}^{r} C_{j,r}\}$ $+ K_r^2 \{\prod_{j=1}^{r-1} C_{j,r}\} \leq \lambda_r^r \{\prod_{j=1}^{r} \lambda_j\} L(c\lambda_r) \{\sum_{j=1}^{r} \lambda_j + L(c\lambda_r)\} \leq \lambda^{2r} L(c\lambda) \{\lambda + L(c\lambda)\}$, where $m_i = \partial_i(A)$, $n_i = \partial_i(B)$, $\mu_i = m_i + 1$, $\nu_i = n_i + 1$, $\lambda_i = \mu_i + \nu_i$, $C_{j,r} = \mu_j \nu_r + \nu_j \mu_r$, $S_{i,r} = \mu_1 \cdots \mu_i \mu_r + \nu_1 \cdots \nu_i \nu_r$, $a = |A|_1$, $b = |B|_1$, $K_r = \nu_r L(a) + \mu_r L(b) + \lambda_r L(\lambda_r)$, $c = \max(a,b)$ and $\lambda = \max_{1 \leq i \leq r} \lambda_i$.

C = CPRES(p,A,B)

A and B are polynomials over GF(p) in r variables, $r \geq 1$, with $\partial_r(A) \geq \partial_r(B) > 0$. C is the resultant of A and B, a polynomial over GF(p) in r - 1 variables (or, if r = 1, an element of GF(p)).

Algorithm

(1) If TYPE(TAIL(A)) $\neq 0$, go to (2); C ← CPRES1(p,A,B); return.

(2) M ← CPDEGS(A); DECAP(m,M); M ← INV(M); DECAP($m_1$,M); erase M.

(3) N ← CPDEGS(B); DECAP(n,N); N ← INV(N); DECAP($n_1$,N); erase N.

(4) k ← $mn_1 + nm_1$; r ← CPNV(A) - 1; C ← 0; D ← PFA(0,PFA(1,0)); i ← -1.

(5) i ← i + 1; if i = p, stop.

(6) $A^*$ ← CPUEV(p,A,i); if CPDEG($A^*$) < m, (erase $A^*$; go to (5)).

(7) $B^*$ ← CPUEV(p,B,i); if CPDEG($B^*$) < n, (erase $A^*$, $B^*$; go to (5)).

(8) $C^*$ ← CPRES(p,$A^*$,$B^*$); erase $A^*$,$B^*$.

(9) CPINTI(p,C,i,$C^*$,D,r,t); if CPDEG(D) $\leq$ k, go to (5).

(10) Erase D; return.

Computing Time: $\leq \sum_{i=1}^{r-1} \{\prod_{j=1}^{i} C_{j,r}\}\{\prod_{j=1}^{r} \mu_j + \prod_{j=1}^{r} \nu_j\} + \sum_{i=1}^{r} \{\prod_{j=1}^{i} C_{j,r}\}$ $\{\prod_{j=i}^{r-1} \lambda_j\} \leq \{\prod_{i=1}^{r} \kappa_i\} \sum_{i=1}^{r} \kappa_i \kappa_r^i + \sum_{i=1}^{r} \{\prod_{j=1}^{i} \kappa_j\}\{\prod_{j=i}^{r-1} \lambda_j\} \kappa_r^i \leq \kappa_r^{r-1} \{\prod_{j=1}^{r} \kappa_j\}$ $\{\sum_{j=1}^{r} \kappa_j\} \leq \kappa^{2r}$, where $m_i = \partial_i(A)$, $n_i = \partial_i(B)$, $\ell_i = \partial_i(C)$, $\mu_i = m_i + 1$, $\nu_i = n_i + 1$, $\lambda_i = \ell_i + 1$, $C_{j,r} = \mu_j \nu_r + \nu_j \mu_r$, $\kappa_i = \max(\mu_i, \nu_i)$ and $\kappa = \max_{1 \leq i \leq r} \kappa_i$.

c = CPRES1(p,A,B,)

A and B are univariate polynomials over GF(p) with $\deg(A) \geq \deg(B) > 0$. c is the resultant of A and B, an element of GF(p).

Algorithm

Let $A_1, A_2, \ldots, A_k$ be the complete p.r.s. over GF(p) defined by $A_1 = A$, $A_2 = B$ and $A_i = Q_i A_{i+1} + A_{i+2}$ for $1 \leq i \leq k-2$ where $Q_i$ is a polynomial over GF(p) and $\deg(A_{i+2}) < \deg(A_{i+1})$. Let $c_i = \text{ldcf}(A_i)$ and $n_i = \deg(A_i)$. By the fundamental theorem of polynomial remainder sequences,

$$\text{resultant}(A_1, A_2) = (-1)^{\nu} \prod_{i=2}^{k-1} c_i^{n_{i-1} - n_{i+1}} c_k^{n_{k-1}},$$

where $\nu = \sum_{i=1}^{k-2} n_i n_{i+1}$. The following algorithm employs this formula.

(1) $A_1 \leftarrow$ BORROW(A); $A_2 \leftarrow$ BORROW(B); $\nu \leftarrow 0$; $c \leftarrow 1$.

(2) $A_3 \leftarrow$ CPREM(p,$A_1$,$A_2$); if $A_3 = 0$, ($c \leftarrow 0$; go to (9)).

(3) $n_1 \leftarrow \deg(A_1)$; $n_2 \leftarrow \deg(A_2)$; $n_3 \leftarrow \deg(A_3)$.

(4) $m_1 \leftarrow \text{mod}(n_1, 2)$; $m_2 \leftarrow \text{mod}(n_2, 2)$; $\nu \leftarrow \text{mod}(\nu + m_1 m_2, 2)$.

(5) $n \leftarrow n_1 - n_3$; $c_2 \leftarrow \text{ldcf}(A_2)$.

(6) Do for $i = 1,\ldots,n(c \leftarrow \mathrm{CPROD}(p,c,c_2))$.

(7) Erase $A_1$; $A_1 \leftarrow A_2$; $A_2 \leftarrow A_3$; if $n_3 > 0$, go to (2).

(8) $c_2 \leftarrow \mathrm{ldcf}(A_2)$; do for $i = 1,\ldots,n_2(c \leftarrow \mathrm{CPROD}(p,c,c_2))$; if $v \neq 0$, $c \leftarrow p - c$.

(9) Erase $A_1$ and $A_2$; return.

Computing Time: $\leq mn$, where $m = \deg(A)$ and $n = \deg(B)$.

## Polynomial Multiplication and Division

C = PMPY(A,B)

A and B are polynomials with integer coefficients in r variables, $r \geq 1$. $C = A \cdot B$.

Algorithm

(1) $C \leftarrow 0$; if $A = 0$ or $B = 0$, return.

(2) $d \leftarrow \mathrm{PNORMF}(A)$; $e \leftarrow \mathrm{PNORMF}(B)$; $f \leftarrow \mathrm{IPROD}(d,e)$; erase d,e; IMFI(f,2).

(3) $I \leftarrow$ PRIME; $L \leftarrow \mathrm{INV}(\mathrm{PVLIST}(A))$; $Q \leftarrow$ PFA(1,0).

(4) If $I = 0$, stop; ADV(p,I); $A^* \leftarrow \mathrm{CPMOD}(p,A)$; $B^* \leftarrow \mathrm{CPMOD}(p,B)$.

(5) $C^* \leftarrow \mathrm{CPMPY}(p,A^*,B^*)$; erase $A^*$, $B^*$.

(6) $q \leftarrow \mathrm{CRECIP}(p,\mathrm{CMOD}(p,Q))$; $C_1 \leftarrow$ CPCRA (Q,C,p,C*,q,L); erase $C^*$ and C; $C \leftarrow C_1$.

(7) IMFI(Q,p); if $\mathrm{ICOMP}(Q,f) < 0$, go to (4).

(8) Erase L, f and Q; return.

Computing Time: $\leq (\alpha + \beta)(L\alpha + M\beta + N\gamma + N\nu) \leq N(\alpha + \beta)(\alpha + \beta + \nu)$, where $\ell_i = \partial_i(A)$, $m_i = \partial_i(B)$, $n_i = \partial_i(C)$, $\lambda_i = \ell_i + 1$, $\mu_i = m_i + 1$, $\nu_i = n_i + 1$, $L = \Pi_{i=1}^{r} \lambda_i$, $M = \Pi_{i=1}^{r} \mu_i$, $N = \Pi_{i=1}^{r} \nu_i$, $a = |A|_1$, $b = |B|_1$, $c = |C|_1$, $\alpha = L(a)$, $\beta = L(b)$, $\gamma = L(c)$ and $\nu = \max_{1 \leq i \leq r} \nu_i$.

C = PDIV(A,B)

A and B are polynomials with integer coefficients in r variables, $r \geq 1$, such that $B \neq 0$ and A is divisible by B. C = A/B.

Algorithm

(1) $C \leftarrow 0$; if $A = 0$, return.

(2) $Q \leftarrow$ PNORMF(A); ELPOF2(Q,T,h); erase Q; $Q \leftarrow$ PNORMF(B); ELPOF2(Q,T,k); erase Q; $f \leftarrow$ PEXP; $m \leftarrow 0$; $n \leftarrow 0$; $Q \leftarrow$ PFA(1,0); $I \leftarrow$ PRIME; $L \leftarrow$ INV(PVLIST(A)).

(3) If $I = 0$, stop; ADV(p,I); $B^* \leftarrow$ CPMOD(p,B); if $B^* = 0$, go to (3).

(4) $A^* \leftarrow$ CPMOD(p,A); $C^* \leftarrow$ CPDIV(p,$A^*$,$B^*$); erase $A^*$, $B^*$.

(5) $q \leftarrow$ CRECIP(p,CMOD(p,Q)); $C_1 \leftarrow$ CPCRA(Q,C,p,$C^*$,q,L); $S \leftarrow$ PCOMP(C,$C_1$); erase C, $C^*$; $C \leftarrow C_1$; IMPI(Q,p); $m \leftarrow m + 1$; if $S \neq 0$, ($n \leftarrow m$; go to (3)); $g \leftarrow m \cdot f$; if $g \leq h$ or $g < k + n(f + 1)$, go to (3).

(6) Erase L, Q; return.

Computing Time: $\leq (L\alpha + M\beta + N\gamma + L\lambda)\,\theta \leq L\lambda\theta + L\theta^2$, where $a = |A|_1$, $b = |B|_1$, $c = |C|_\infty$, $\alpha = L(a)$, $\beta = L(b)$, $\gamma = L(c)$, $\ell_i = \partial_i(A)$, $m_i = \partial_i(B)$, $n_i = \partial_i(C)$, $\lambda_i = \ell_i + 1$, $\mu_i = m_i + 1$, $\nu_i = n_i + 1$, $L = \Pi_{i=1}^{r} \lambda_i$, $M = \Pi_{i=1}^{r} \mu_i$, $N = \Pi_{i=1}^{r} \nu_i$, $\lambda = \max_{1\leq i\leq r} \lambda_i$, and $\theta = \alpha + \beta + \gamma$.

C = PTDIV(A,B)

A and B are polynomials with integer coefficients in r variables, $r \geq 1$. If $B \neq 0$ and A is divisible by B, then C = A/B. Otherwise, C = -1, a Fortran integer.

Algorithm

(1) $C \leftarrow -1$; if $B = 0$, return; $C \leftarrow 0$; if $A = 0$, return.

(2) $Q \leftarrow$ PNORMF(A); ELPOF2(Q,T,h); erase Q; $Q \leftarrow$ PNORMF(B); ELPOF2(Q,T,k); erase Q; $f \leftarrow$ PEXP; $m \leftarrow 0$; $n \leftarrow 0$; $Q \leftarrow$ PFA(1,0); $I \leftarrow$ PRIME; $L \leftarrow$ INV(PVLIST(A)).

(3) If $I = 0$, stop; ADV(p,I); $B^* \leftarrow$ CPMOD(p,B); if $B^* = 0$, go to (3).

(4) $A^* \leftarrow$ CPMOD(p,A); $C^* \leftarrow$ CPTDIV(p,$A^*$,$B^*$); erase $A^*$,$B^*$; if $C^* = -1$, go to (6).

(5) $q \leftarrow$ CRECIP(p,CMOD(p,Q)); $C_1 \leftarrow$ CPCRA(Q,C,p,$C^*$,q,L); $S \leftarrow$ PCOMP(C,$C_1$); erase C, $C^*$; $C \leftarrow C_1$; IMFI(Q,p); $m \leftarrow m + 1$; if $S \neq 0$, ($n \leftarrow m$; go to (3)); $g \leftarrow m \circ f$; if $g \leq h$ or $g < k + n(f + 1)$, go to (3); go to (7).

(6) Erase C; $C \leftarrow -1$.

(7) Erase L, Q; return.

Computing Time: $\leq (L\alpha + M\beta + N\gamma + L\lambda)\theta \leq L\lambda\theta + L\theta^2$, where $a = |A|_1$, $b = |B|_1$, $c = |C|_\infty$, $\alpha = L(a)$, $\beta = L(b)$, $\gamma = L(c)$, $\ell_i = \partial_i(A)$, $m_i = \partial_i(B)$, $n_i = \partial_i(C)$, $\lambda_i = \ell_i + 1$, $\mu_i = m_i + 1$, $\nu_i = n_i + 1$, $L = \Pi_{i=1}^{r} \lambda_i$, $M = \Pi_{i=1}^{r} \mu_i$, $N = \Pi_{i=1}^{r} \nu_i$, $\lambda = \max_{1\leq i\leq r} \lambda_i$, and $\theta = \alpha + \beta + \gamma$.

B = PID(A, I)

A is a polynomial with integer coefficients in r variables, $r \geq 0$. I is an infinite-precision integer which is a divisor of A. B is the quotient A/I.

Algorithm

(1) B ← 0; if A = 0, return; if FIRST(I) = 1 and TAIL(I) = 0, (B ← BORROW(A); return); B ← PFL(PVBL(A), 0); L ← TAIL(A).

(2) ADV(C, L); T ← TYPE(C); if T = 0, D ← IQ(C, I); if T = 1, D ← PID(C, I); ADV(E, L); B ← PFA(E, PFL(D, B)); if $L \neq 0$, go to (2).

(3) B ← INV(B); return.

Computing Time: The maximum computing time is $\sim L(b)\ L(I)\ \Pi_{i=1}^{r} \nu_i$, where $b = |B|_\infty$, $n_i = \partial_i(B)$ and $\nu_i = n_i + 1$.

C = CPMPY(p,A,B)

A and B are polynomials over GF(p) in r variables, $r \geq 1$. C = A•B, the product of A and B. A modular algorithm is used.

Algorithm

(1) C ← 0; if A = 0 or B = 0, return.

(2) r ← CPNV(A); if r > 1, go to (3); C ← CPPROD(p,A,B); return.

(3) M ← INV(CPDEGS(A)); DECAP(m,M); erase M; N ← INV(CPDEGS(B)); DECAP(n,N); erase N; k ← m + n; D ← PFA(0,PFA(1,0)); i ← 0.

(4) $A^* \leftarrow$ CPUEV(p,A,i); $B^* \leftarrow$ CPUEV(p,B,i).

(5) $C^* \leftarrow$ CPMPY(p,$A^*$,$B^*$); erase $A^*$,$B^*$.

(6) CPINTI(p,C,i,$C^*$,D,r,t).

(7) If $i \neq k$, (i ← i + 1; go to (4)).

(8) Erase D; return.

Computing Time: $\leq \{\Pi_{i=1}^{r-1} \lambda_i\}\{\mu_r \nu_r + \lambda_r \Sigma_{i=1}^{r-1} \lambda_i\} \leq \{\Pi_{i=1}^{r} \lambda_i\}\{\Sigma_{i=1}^{r} \lambda_i\} \leq \lambda^{r+1}$, where $m_i = \partial_i(A)$, $n_i = \partial_i(B)$, $\mu_i = m_i + 1$, $\nu_i = n_i + 1$, $\lambda_i = \mu_i + \nu_i$, and $\lambda = \max_{1 \leq i \leq r} \lambda_i$.

C = CPDIV(p,A,B)

A and B are polynomials over GF(p) in r variables, $r \geq 1$, such that $B \neq 0$ and A is divisible by B. C = A/B.

Algorithm

(1) $C \leftarrow 0$; if $A = 0$, return.

(2) $r \leftarrow$ CPNV(A); $L \leftarrow$ INV(CPDEGS(A)); DECAP($\ell$,L); erase L; $M \leftarrow$ INV(CPDEGS(B)); DECAP(m,M); erase M; $n \leftarrow \ell - m$; if $r > 1$, go to (4).

(3) $T \leftarrow$ CPQREM(p,A,B,); DECAP(C,T); DECAP(R,T); return.

(4) $i \leftarrow -1$; $D \leftarrow$ PFA(0,PFA(1,0)).

(5) $i \leftarrow i + 1$; $B^* \leftarrow$ CPUEV(p,B,i); if $B^* = 0$, go to (5).

(6) $A^* \leftarrow$ CPUEV(p,A,i).

(7) $C^* \leftarrow$ CPDIV(p,$A^*$,$B^*$); erase $A^*$,$B^*$.

(8) CPINTI(p,C,i,$C^*$,D,r,t).

(9) If $n > 0$, ($n \leftarrow n - 1$; go to (5)).

(10) Erase D; return.

Computing Time: $\leq \lambda_1\{\Pi_{j=1}^{r} u_j\} + u_r\{\Pi_{j=1}^{r} \nu_j\} + \Sigma_{i=1}^{r-1}\{\Pi_{j=1}^{i} \nu_j\}\{\Pi_{j=i}^{r} \lambda_j\} + \Sigma_{i=2}^{r-1}\{\Pi_{j=1}^{i-1} \nu_j\}\lambda_i\{\Pi_{j=i}^{r} u_j\} \leq \{\Sigma_{i=1}^{r} \lambda_i\}\{\Pi_{i=1}^{r} \lambda_i\} \leq \lambda^{r+1}$, where $\ell_i = \partial_i(A)$, $m_i = \partial_i(B)$, $n_i = \partial_i(C)$, $\lambda_i = \ell_i + 1$, $u_i = m_i + 1$, $\nu_i = n_i + 1$ and $\lambda = \max_{1 \leq i \leq r} \lambda_i$.

C = CPTDIV(p,A,B)

A and B are polynomials over GF(p) in r variables, $r \geq 1$. If $B \neq 0$ and A is divisible by B, then C = A/B. Otherwise C = -1, a Fortran integer.

Algorithm

(1) $C \leftarrow -1$; if $B = 0$, return; $C \leftarrow 0$; if $A = 0$, return.

(2) $T \leftarrow$ CPDEGS(A); $U \leftarrow$ CPDEGS(B).

(3) DECAP($\ell$,T); DECAP(m,U); $n \leftarrow \ell - m$; if $n < 0$, $C \leftarrow -1$; if $T \neq 0$, go to (3).

(4) If $C = -1$, return; $r \leftarrow$ CPNV(A); if $r > 1$, go to (6).

(5) $T \leftarrow$ CPQREM(p,A,B); DECAP(C,T); DECAP(R,T); if $R = 0$, return; erase C,R; $C \leftarrow -1$; return.

(6) $i \leftarrow -1$; $j \leftarrow 0$; $D \leftarrow$ PFA(0,PFA(1,0)).

(7) $i \leftarrow i + 1$; if $i = p$, stop; $A^* \leftarrow$ CPUEV(p,A,i); $B^* \leftarrow$ CPUEV(p,B,i); if $A^* = 0$ and $B^* = 0$, go to (7).

(8) $C^* \leftarrow$ CPTDIV(p,$A^*$,$B^*$); erase $A^*$,$B^*$; if $C^* = -1$, go to (11).

(9) CPINTI(p,C,i,$C^*$,D,r,t); $j \leftarrow j + 1$.

(10) If CPDEG(C) > n, go to (11); if $j \leq \ell$, go to (7); go to (12).

(11) Erase C; $C \leftarrow -1$.

(12) Erase D; return.

Computing Time: For $r = 1$, $\sim 1$ if $\ell_1 < m_1$ and $\sim \mu_1 \nu_1$ if $\ell_1 \geq m_1$. For $r > 1$, $\leq \Pi_{i=2}^{r} \lambda_i + \Pi_{i=2}^{r} \mu_i$ if $\ell_i < m_i$ for some i, $\leq \{\Pi_{i=1}^{r} \lambda_i\}\{\Sigma_{i=1}^{r} \lambda_i + \mu_r\} \leq \{\Pi_{i=1}^{r} \lambda_i\}$ $\{\Sigma_{i=1}^{r} \lambda_i\} \leq \lambda^{r+1}$ otherwise, where $\ell_i = \partial_i(A)$, $m_i = \partial_i(B)$, $\lambda_i = \ell_i + 1$, $\mu_i = m_i + 1$ and $\nu_1 = \ell_1 - m_1 + 1$.

C = CPQ(p,A,B)

p is a prime number. A and B are polynomials over GF(p) in r variables, $r \geq 1$. If $B \neq 0$ and $B|A$, then $C = A/B$. Otherwise, C is the Fortran integer -1. The "classical" polynomial division algorithm is used.

Algorithm

(1) $C \leftarrow -1$; if $B = 0$, return; $C \leftarrow 0$; if $A = 0$, return.

(2) $M \leftarrow$ CPDEGS(A); $N \leftarrow$ CPDEGS(B); $L \leftarrow 0$; $h \leftarrow 0$.

(3) DECAP(m,M); DECAP(n,N); $\ell \leftarrow m - n$; $L \leftarrow$ PFA($\ell$,L); if $\ell < 0$, $h \leftarrow \ell$; if $M \neq 0$, go to (3).

(4) $L \leftarrow$ INV(L); if $h < 0$, ($C \leftarrow -1$; erase L; return).

(5) $C \leftarrow$ CPQDB(p,A,B,L); erase L; return.

Computing Time: $\leq \prod_{i=1}^{r} \lambda_i \nu_i$ if $m_i \geq n_i$ for all i, $\leq \prod_{i=2}^{r} \mu_i + \prod_{i=2}^{r} \nu_i$ otherwise, where $m_i = \partial_i(A)$, $n_i = \partial_i(B)$, $\ell_i = m_i - n_i$, $\mu_i = m_i + 1$, $\nu_i = n_i + 1$ and $\lambda_i = \ell_i + 1$.

C = CPQDB(p,A,B,L)

p is a prime number. A and B are polynomials over GF(p) in r variables, $r \geq 1$. L is a list $(\ell_r,\ldots,\ell_1)$ of non-negative integers. If $B \neq 0$, $B \mid A$ and $\partial_i(A/B) \leq \ell_i$ for $1 \leq i \leq r$, then $C = A/B$. Otherwise, C is the Fortran integer -1. The "classical" polynomial division algorithm is used.

Algorithm

(1) $C \leftarrow -1$; if $B = 0$, return; $C \leftarrow 0$; if $A = 0$, return.

(2) $m \leftarrow$ CPDEG(A); $n \leftarrow$ CPDEG(B); $h \leftarrow m - n$; if $h < 0$ or $h >$ FIRST(L), ($C \leftarrow -1$; return).

(3) $L^* \leftarrow$ TAIL(L); if $L^* \neq 0$, go to (5).

(4) $T \leftarrow$ CPQREM(p,A,B); DECAP(C,T); DECAP(R,T); if $R = 0$, return; erase C,R; $C \leftarrow -1$; return.

(5) $\bar{A} \leftarrow$ TAIL(A); $\bar{B} \leftarrow$ TAIL(B); $\bar{A} \leftarrow$ INV(CINV($\bar{A}$)); ADV(F,$\bar{B}$); $C \leftarrow$ PFA(h,0).

(6) DECAP(E,$\bar{A}$); $G \leftarrow$ CPQDB(p,E,F,$L^*$); erase E; if $G < 0$, go to (10); $C \leftarrow$ PFL(G,C); if $G = 0$, go to (8); $S \leftarrow \bar{A}$; $T \leftarrow \bar{B}$.

(7) If $T = 0$, go to (8); ADV(W,T); $Z \leftarrow$ CPPROD(p,W,G); $V \leftarrow$ CPDIF(p,FIRST(S),Z); erase FIRST(S); ALTER(V,S); erase Z; $S \leftarrow$ TAIL(S); go to (7).

(8) $h \leftarrow h - 1$; if $h \geq 0$, go to (6).

(9) $C \leftarrow$ INV(C).

(10) If $\bar{A} = 0$, return; if FIRST($\bar{A}$) $\neq 0$, go to (11); DECAP(W,$\bar{A}$); go to (10).

(11) DECAP(W,$\bar{A}$); erase W; if $\bar{A} \neq 0$, go to (11).

(12) Erase C; $C \leftarrow -1$; return.

Computing Time: $\leq \sum_{i=1}^{r} \{\prod_{j=1}^{i-1} (\lambda_j + \nu_j + \mu_j)\}\{\prod_{j=i}^{r} \lambda_j \nu_j\}$, where $L = (\ell_r,\ldots,\ell_1)$, $n_i = \partial_i(B)$, $m_i = \partial_i(A)$, $\lambda_i = \ell_i + 1$, $\mu_i = m_i + 1$ and $\nu_i = n_i + 1$.

C = CPMPYI(p,A,B,r)

A is a polynomial over GF(p) in r - 1 variables, $r \geq 1$, and B is a univariate polynomial over GF(p). C is the polynomial over GF(p) in r variables defined by $C(x_1,\ldots,x_r) = B(x_1) \cdot A(x_2,\ldots,x_r)$.

Algorithm

(1) C ← 0; if A = 0 or B = 0, return.

(2) If r = 1, (C ← CSPROD(p,B,A,0); return).

(3) T ← A; ADV(n,T); C ← PFA(n,C).

(4) ADV(d,T); E ← CPMPYI(p,d,B,r-1); C ← PFL(E,C); if $T \neq 0$, go to (4).

(5) C ← INV(C); return.

Computing Time: The maximum computing time is $\sim \prod_{i=1}^{r} \nu_i$, where $n_i = \partial_i(A)$ and $\nu_i = n_i + 1$.

C = CPUPR(p,A,B)

A is a non-zero polynomial $A(x_1,\ldots,x_r)$ over GF(p) in r variables, $r \geq 1$. $B = B(x_1)$ is a non-zero univariate polynomial over GF(p). $C = A(x_1,\ldots,x_r) \cdot B(x_1)$, the product of A and B.

Algorithm

(1) If TYPE(TAIL(A)) = 0, (C ← CPPROD(p,A,B); return); n ← FIRST(A); C ← PFA(n,0); L ← TAIL(A).

(2) ADV(D,L); E ← CPUPR(p,D,B); C ← PFL(E,C); if $L \neq 0$, go to (2).

(3) C ← INV(C); return.

Computing Time: The maximum computing time is $\sim \nu_1 \prod_{i=1}^{r} \mu_i$, where $m_i = \partial_i(A)$, $n_1 = \partial_1(B)$, $\mu_i = m_i + 1$ and $\nu_1 = n_1 + 1$.

C = CPUDIV(p,A,B)

A is a non-zero polynomial over GF(p) in r variables, $r \geq 1$, $A = A(x_1,\ldots,x_r)$, where $x_r$ is the main variable. B is a univariate polynomial over GF(p), $B(x_1)$, which is a divisor of A. C is the quotient $C(x_1,\ldots,x_r) = A(x_1,\ldots,x_r)/B(x_1)$.

Algorithm

(1) If TYPE(TAIL(A)) = 0, (C ← CPQ(p,A,B); return); n ← FIRST(A); C ← PFA(n,0); L ← TAIL(A).

(2) ADV(D,L); E ← CPUDIV(p,D,B); C ← PFL(E,C); if $L \neq 0$, go to (2).

(3) C ← INV(C); return.

Computing Time: The maximum computing time is $\sim \mu_1 \{\prod_{i=1}^{r} \nu_i\}$, where $m_1 = \partial_1(B)$, $n_i = \partial_i(C)$, $\mu_1 = m_1 + 1$ and $\nu_i = n_i + 1$.

## Chinese Remainder Algorithm

C = CPCRA(Q,B,p,A,q,L)

p is an odd prime number and A is a polynomial over GF(p) in r variables, $r \geq 0$. Q is an odd positive infinite-precision integer relatively prime to p and B is a polynomial $B(x_1,\ldots,x_r)$ whose integer coefficients are all less than Q/2 in absolute value. q is an element of GF(p), $0 < q < p$, such that $qQ \equiv 1$ (modulo p). L is the list of variables $(x_r,\ldots,x_1)$. C is the unique polynomial, with infinite-precision integer coefficients less than pQ/2 in absolute value, such that $C \equiv B$ (modulo Q) and $C \equiv A$ (modulo p). Note that if $Q = 1$, and hence $B = 0$, then $C \equiv A$ (modulo p) and the coefficients of C are less than p/2 in absolute value.

Algorithm

(1) If L = 0, (C ← CCRA(Q,B,p,A,q); return).

(2) C ← 0; m ← -1; $\bar{A} \leftarrow A$; if $\bar{A} \neq 0$, ADV(m,$\bar{A}$); n ← -1; $\bar{B} \leftarrow B$; if $\bar{B} \neq 0$, ($\bar{B} \leftarrow$ TAIL($\bar{B}$); n ← FIRST(TAIL($\bar{B}$))); M ← TAIL(L).

(3) If $\bar{A} = 0$ and $\bar{B} = 0$, go to (6); E ← 0; F ← 0; S ← m - n; if $S \geq 0$, (k ← m; m ← m - 1; ADV(E,$\bar{A}$)); if $S \leq 0$, (k ← n; n ← -1; ADV(F,$\bar{B}$); $\bar{B} \leftarrow$ TAIL($\bar{B}$); if $\bar{B} \neq 0$, n ← FIRST(TAIL($\bar{B}$))); if E = 0 and F = 0, go to (3).

(4) G ← CPCRA(Q,F,p,E,q,M).

(5) C ← PFA(k, PFL(G,C)); go to (3).

(6) If $C \neq 0$, C ← PFL(BORROW(FIRST(L)), INV(C)); return.

Computing Time: $\leq \{\Pi_{i=1}^{r} \mu_i\} + \lambda \leq L(c)\{\Pi_{i=1}^{r} \mu_i\}$, where $m_i = \partial_i(C)$, $\mu_i = m_i + 1$, $\lambda$ is the sum of the lengths of the non-zero integer coefficients of $C$, and $c = |C|_\infty$.

$$C = CCRA(Q,B,p,a,q)$$

$p$ is an odd prime number and $a$ is an element of $GF(p)$. $Q$ is an odd positive infinite-precision integer which is relatively prime to $p$, and $B$ is an infinite-precision integer such that $|B| < Q/2$. $q$ is an element of $GF(p)$, $0 < q < p$, such that $qQ \equiv 1$ (modulo p). $C$ is the unique infinite-precision integer such that $C \equiv B$ (modulo Q), $C \equiv a$ (modulo p), and $|C| < pQ/2$. Note that if $Q = 1$, and hence $B = 0$, then $C \equiv a$ (modulo p) and $|C| < p/2$.

Algorithm

(1) $b \leftarrow CMOD(p,B)$; $c \leftarrow CDIF(p,a,b)$; if $c = 0$, ($C \leftarrow BORROW(B)$; return).

(2) $d \leftarrow CPROD(p,c,q)$; if $d + d > p$, $d \leftarrow d - p$.

(3) $D \leftarrow PFA(d,0)$; $T \leftarrow IPROD(D,Q)$; erase D; $C \leftarrow ISUM(T,B)$; erase T; return.

Computing Time: $\sim L(C)$.

## Evaluation and Interpolation

B = CPUEV(p,A,e)

A is a polynomial over GF(p) in r variables, $r \geq 1$, and e is an element of GF(p). $B = B(x_2,\ldots,x_r) = A(e,x_2,\ldots,x_r)$.

Algorithm

(1) $B \leftarrow 0$; if $A = 0$, return; $r \leftarrow$ CPNV(A); if $r = 1$, ($B \leftarrow$ CPEVAL(p,A,e); return); $n \leftarrow$ CPDEG(A); $\bar{A} \leftarrow$ TAIL(A).

(2) ADV(C,$\bar{A}$); if $r > 2$, go to (3); $D \leftarrow$ CPEVAL(p,C,e); if $B = 0$ and $D = 0$, go to (4); if $B = 0$, $B \leftarrow$ PFA(n,B); $B \leftarrow$ PFA(D,B).

(3) $D \leftarrow$ CPUEV(p,C,e); if $B = 0$ and $D = 0$, go to (4); If $B = 0$, $B \leftarrow$ PFA(n,B); $B \leftarrow$ PFL(D,B).

(4) $n \leftarrow n - 1$; if $\bar{A} \neq 0$, go to (2); $B \leftarrow$ INV(B); return.

Computing Time: The maximum computing time is $\sim \Pi_{i=1}^{r} \nu_i$, where $n_i = \partial_i(A)$ and $\nu_i = n_i + 1$.

CPINTI(p,A,b,C,D,r,t)

A is a polynomial over GF(p) in r variables, $r \geq 1$, C is a polynomial over GF(p) in r - 1 variables, D is a univariate polynomial over GF(p), and b is an element of GF(p). D is a product $D(x_1) = \Pi_{i=1}^{k}(x_1 - b_i)$ where b and the $b_i$ are distinct elements of GF(p), or else $k = 0$ and $D(x_1) = 1$. Either $A = 0$ or $\partial_1(A) < \partial_1(D) = k$.

Let $G$ be the unique polynomial over $GF(p)$ such that $G(b_i,x_2,\ldots,x_r) = A(b_i,x_2,\ldots,x_r)$ for $1 \leq i \leq k$, $G(b,x_2,\ldots,x_r) = C(x_2,\ldots,x_r)$, and $\partial_1(G) \leq k$. $A$ is erased and $G$ is returned as the new value of $A$. Let $\bar{D}(x_1) = D(x_1) \cdot (x_1-b)$. $D$ is erased and $\bar{D}$ is returned as the new value of $D$. If $G = A$, then $t = 0$; otherwise $t = 1$, a Fortran integer.

Algorithm

(1) $A^* \leftarrow$ CPUEV(p,A,b).

(2) If $r = 1$, ($U \leftarrow$ CDIF(p,C,$A^*$); go to (3)); $U \leftarrow$ CPDIF(p,C,$A^*$); erase C, $A^*$.

(3) $t \leftarrow 1$; if $U = 0$, ($t \leftarrow 0$; go to (5)).

(4) $v \leftarrow$ CPEVAL(p,D,b); $v \leftarrow$ CRECIP(p,V); $E \leftarrow$ CSPROD(p,D,v,0); $W \leftarrow$ CPMPYI(p,U,E,r); erase E; if $r > 1$, erase U; $G \leftarrow$ CPSUM(p,W,A); erase A,W; $A \leftarrow G$.

(5) $U \leftarrow$ PFA(1,PFA(1,PFA(CDIF(p,0,b),0))); $W \leftarrow$ CPPROD(p,D,U); erase D,U; $D \leftarrow W$; return.

Computing Time: $\leq \{\prod_{i=1}^{r} \nu_i\} + k + 1$, where $n_i = \partial_i(G)$, $\nu_i = n_i + 1$ and $k = \partial_1(D)$. Note that $n_1 = \partial_1(A)$ if $G = A$, $n_1 = \partial_1(D)$ if $G \neq A$, and $n_i = \max(\partial_i(A), \partial_i(C))$ for $2 \leq i \leq r$.

## Degrees and Leading Coefficients

L = PDEGS(A)

A is a polynomial with integer coefficients in r variables, $r \geq 1$. If $A = 0$, then L is the null list. Otherwise, $L = (n_r, \ldots, n_1)$ where $n_i$ is the degree of $A(x_1, \ldots, x_r)$ in $x_i$ and $x_r$ is the main variable. Each $n_i$ is a Fortran integer.

Algorithm

(1) $L \leftarrow 0$; if $A = 0$, return; $N \leftarrow$ PDEG(A); $T \leftarrow$ TAIL(A); $M \leftarrow$ FIRST(T).

(2) If TYPE(M) $\neq 0$, go to (4).

(3) $L \leftarrow$ PFA(N, 0); return.

(4) $L \leftarrow$ PFA(0, L); $M \leftarrow$ FIRST(TAIL(M)); if TYPE(M) $\neq 0$, go to (4).

(5) $C \leftarrow$ FIRST(T); $K \leftarrow$ PDEGS(C); $V \leftarrow L$.

(6) DECAP(S, K); if S > FIRST(V), ALTER(S, V); $V \leftarrow$ TAIL(V); if $V \neq 0$, go to (6).

(7) $T \leftarrow$ TAIL(TAIL(T)); if $T \neq 0$, go to (5).

(8) $L \leftarrow$ PFA(N, L); return.

Computing Time: The maximum computing time is $\sim \prod_{i=2}^{r} \nu_i$, where $n_i = \partial_i(A)$ and $\nu_i = n_i + 1$.

I = PLNCF(A)

A is a polynomial with integer coefficients in r variables, $r \geq 0$. If A = 0, or if r = 0, then I = A. Otherwise, I is the leading numerical coefficient of A.

Algorithm

(1) If A = 0, (I ← 0; return); B ← A; go to (3).

(2) B ← TAIL(FIRST(B)).

(3) If TYPE(B) ≠ 0, go to (2); I ← BORROW(B); return.

Computing Time: $\sim r + 1$.

L = CPDEGS(A)

A is a non-zero polynomial over a field GF(p) in r variables, $r \geq 1$. Let $A = A(x_1,\ldots,x_r)$, where $x_r$ is the main variable. Let $n_i$ be the degree of A in $x_i$. Then $L = (n_r,\ldots,n_1)$, a list of Fortran integers.

Algorithm

(1) N ← FIRST(A); T ← TAIL(A); if TYPE(T) ≠ 0, go to (2); L ← PFA(N,0); return.

(2) ADV(B,T); L ← CPDEGS(B).

(3) If T = 0, go to (5); ADV(B,T); if B = 0, go to (3); M ← CPDEGS(B); U ← L; V ← M.

(4) $h \leftarrow$ FIRST(U); DECAP(k,V); if $h < k$, ALTER(k,U); $U \leftarrow$ TAIL(U); if $U \neq 0$, go to (4); go to (3).

(5) $L \leftarrow$ PFA(N,L); return.

Computing Time: The maximum computing time is $\sim \prod_{i=2}^{r} \nu_i$, where $\nu_i = n_i + 1$.

M = CPDEG(A)

A is a polynomial over a field GF(p) in r variables, $r \geq 1$. M is the degree of A in its main variable, a Fortran integer (if $A = 0$, then $M = 0$).

Algorithm

(1) $M \leftarrow 0$.

(2) If $A \neq 0$, $M \leftarrow$ FIRST(A).

(3) Return.

Computing Time: $\sim 1$.

$n_1$ = CPDEG1(A)

A is a non-zero polynomial $A(x_1,\ldots,x_r)$ in r variables, $r \geq 1$, over a field GF(p). $n_1$ is the degree of A in $x_1$.

Algorithm

(1) L ← INV(CPDEGS(A)); $n_1$ ← FIRST(L); erase L; return.

B = CPLNCF(A)

A is a non-zero polynomial over a field GF(p) in r variables, $r \geq 1$. B is the leading numerical coefficient of A, a non-zero element of GF(p).

Algorithm

(1) B ← A.

(2) B ← TAIL(B); T ← TYPE(B); B ← FIRST(B); if $T \neq 0$, go to (2); return.

Computing Time: $\sim r$.

B = CPLUCF(A)

A is a non-zero polynomial over a field GF(p) in r variables, $r \geq 2$. B is the leading univariate coefficient of A, defined as follows. If r = 2, then B is the leading coefficient of A; if r > 2, then B is the leading univariate coefficient of the leading coefficient of A.

Algorithm

(1) $B \leftarrow A$.

(2) $C \leftarrow$ FIRST(TAIL(B)); if TYPE(TAIL(C)) = 0, ($B \leftarrow$ BORROW(C); return); $B \leftarrow C$; go to (2).

Computing Time: $\sim r$.

## Polynomial Comparison

T = PCOMP(A,B)

A and B are compatible polynomials with integer coefficients in r variables, $r \geq 0$. T is a Fortran integer. T = 0 if A = B and T = 1 if $A \neq B$.

Algorithm

(1) If $A \neq 0$, go to (2); if $B \neq 0$, go to (10); go to (9).

(2) If B = 0, go to (10); if TYPE(A) $\neq 0$, go to (5).

(3) ADV(E,A); ADV(F,B); if $E \neq F$, go to (10); if A = 0, go to (4); if $B \neq 0$, go to (3); go to (10).

(4) If B = 0, go to (9); go to (10).

(5) $A \leftarrow$ TAIL(A); $B \leftarrow$ TAIL(B).

(6) ADV(E,A); ADV(F,B); $T \leftarrow$ PCOMP(E,F).

(7) If T = 1, return; ADV(E,A); ADV(F,B); if $E \neq F$, go to (10); if A = 0, go to (8); if $B \neq 0$, go to (6); go to (10).

(8) If $B \neq 0$, go to (10).

(9) T = 0; return.

(10) T = 1; return.

Computing Time: $\leq \min(L(a), L(b)) \cdot \prod_{i=1}^{r} \min(\mu_i, \nu_i)$, where $a = |A|_\infty$, $b = |B|_\infty$, $m_i = \partial_i(A)$, $n_i = \partial_i(B)$, $\mu_i = m_i + 1$ and $\nu_i = n_i + 1$.

T = CPCOMP(A,B)

A and B are polynomials over a field GF(p) in r variables, $r \geq 1$. T = 0 if A = B and T = 1 if $A \neq B$. T is a Fortran integer.

Algorithm

(1) $T \leftarrow 1$; if $A \neq 0$, go to 2; if $B \neq 0$, return; $T \leftarrow 0$; return.

(2) If B = 0, return; ADV(m,A); ADV(n,B); if $m \neq n$, return;
$U \leftarrow$ TYPE(A).

(3) ADV(C,A); ADV(D,B); if $U \neq 0$, go to (4); if $C \neq D$, return; go to (5).

(4) $T \leftarrow$ CPCOMP(C,D); if $T \neq 0$, return.

(5) If $A \neq 0$, go to (3); $T \leftarrow 0$; return.

Computing Time: $\leq \prod_{i=1}^{r} \min(\mu_i, \nu_i)$, where $m_i = \partial_i(A)$, $n_i = \partial_i(B)$, $\mu_i = m_i + 1$ and $\nu_i = n_i + 1$.

## Miscellaneous

N = PMNORM(A)

A is a non-zero polynomial with integer coefficients in r variables, $r \geq 1$. If $A(x_1,\ldots,x_r) = \sum_{i=0}^{n} A_i(x_1,\ldots,x_{r-1}) \cdot x_r^i$, where $x_r$ is the main variable, then $N = \max_{0\leq i\leq n} \text{norm}(A_i)$.

Algorithm

(1) $N \leftarrow 0$; $T \leftarrow$ TAIL(A).

(2) ADV(C,T); $M \leftarrow$ PNORMF(C); $S \leftarrow$ ICOMP(M,N); if $S > 0$, go to (3); erase M; go to (4).

(3) Erase N; $N \leftarrow M$.

(4) $T \leftarrow$ TAIL(T); if $T \neq 0$, go to (2); return.

Computing Time: $\leq L(N) \cdot \prod_{i=1}^{r} \nu_i$, where $n_i = \partial_i(A)$ and $\nu_i = n_i + 1$.

B = CPMON(p,A)

p is a prime number and A is a non-zero polynomial over GF(p) in r variables, $r \geq 1$. B = A/lncf(A), the associate of A whose heading numerical coefficient is 1.

Algorithm

(1) $b \leftarrow$ CPLNCF(A); $c \leftarrow$ CRECIP(p,B); $B \leftarrow$ CSPROD(p,A,c,0); return.

Computing Time: The maximum computing time is $\sim \prod_{i=1}^{r} \nu_i$, where $n_i = \partial_i(A)$ and $\nu_i = n_i + 1$.

N = CPNV(A)

A is a non-zero polynomial over some field GF(p) in r variables, $r \geq 1$. N is the number of variables, r, a Fortran integer.

Algorithm

(1) $N \leftarrow 1$; $B \leftarrow A$.

(2) $L \leftarrow$ TAIL(B); if TYPE(L) = 0, return.

(3) $B \leftarrow$ FIRST(L); $N \leftarrow N + 1$; go to (2).

Computing Time: $\sim r$.

b = CPONE(A)

A is an arbitrary polynomial over some field GF(p), in r variables, $r \geq 1$. If $A(x_1,\ldots,x_r) = 1$ for all $x_1,\ldots,x_r$, then b = 1; otherwise b = 0. b is a Fortran integer.

Algorithm

(1) If A = 0, go to (4).

(2) If FIRST(A) $\neq$ 0, go to (4); $A \leftarrow$ TAIL(A); if TYPE(A) = 0, go to (3); $A \leftarrow$ FIRST(A); go to (2).

(3) If FIRST(A) $\neq$ 1, go to (4); $b \leftarrow 1$; return.

(4) $B \leftarrow 0$; return.

Computing Time: The maximum computing time is $\sim r$.

L = IFACT(N)

N is a non-negative Fortran integer. L = N!, the factorial of N, an infinite-precision integer.

Algorithm

(1) L ← PFA(1,0); if N < 2, return; I ← 2.

(2) IMFI(L,I); I ← I + 1; if I ≤ N, go to (2); return.

Computing Time: $\leq (N+1)^2$ L(N).

IMFI(A,b)

A is an infinite-precision integer and b is a Fortran integer. The infinite-precision integer C = A · b is computed, A is erased, and A is set to C.

Algorithm

(1) If b ≠ 0, go to (3).

(2) ERLA(A); A ← 0; return.

(3) B ← PFA(b,0); C ← IPROD(A,B); ERLA(A); ERLA(B); A ← C; return.

Computing Time: [illegible] L(A).

## 3. Empirical Computing Times

In the following, empirically observed computing times are given for various algorithms applied to representative sets of inputs. All times given are in seconds and are for the University of Wisconsin SAC-1 implementation on the UNIVAC 1108 computer. This computer has a 36-bit word length and most instructions execute in approximately 0.75 microseconds.

A random polynomial generator was used to obtain inputs to the algorithms for generation of the empirical computing time tables below. Inputs to the random polynomial generator are a degree vector $(n_1,\ldots,n_r)$ and a coefficient size $k$. For every r-tuple $(e_1,\ldots,e_r)$ with $0 \le e_i \le n_i$ the random polynomial generator generates a random integer less than $2^k$ in absolute value, which is the coefficient of $x_1^{e_1} \cdots x_r^{e_r}$ in the random polynomial. For some tables, random polynomials over GF(p) are used. In these cases, where $p$ is approximately $2^{31}$, a random polynomial with, say, 33-bit coefficients is generated and then reduced modulo $p$ using CPMOD of the SAC-1 Modular Arithmetic System.

### Polynomial Greatest Common Divisors

In this category only PGCDCF was timed; times for PGCD would be nearly identical. Table 1 gives times for relatively prime inputs, Table 2 gives times for non-trivial greatest common divisors.

Table 1

The inputs to PGCDCF are random polynomials A and B of degree n in each of one, two or three variables, with 25-bit coefficients. gcd (A,B) = 1 in each case.

| No. of Variables | Degree n | Time | No. of Variables | Degree n | Time | No. of Variables | Degree n | Time |
|---|---|---|---|---|---|---|---|---|
| 1 | 20 | .15 | 2 | 5 | .21 | 3 | 2 | .22 |
| 1 | 40 | .41 | 2 | 10 | .58 | 3 | 4 | .75 |
| 1 | 60 | .63 | 2 | 15 | 1.14 | 3 | 6 | 1.70 |
| 1 | 80 | .97 | 2 | 20 | 1.92 | 3 | 8 | 3.51 |
| 1 | 100 | 1.52 | 2 | 25 | 2.92 | 3 | 10 | 4.84 |

Table 2

The inputs to PGCDCF are $A \cdot C$ and $B \cdot C$, where A, B and C are random polynomials of degree n in each of one, two or three variables, with 25-bit coefficients. $\gcd(A \cdot C, B \cdot C) = C$ in each case.

| No. of Variables | Degree n | Time | No. of Variables | Degree n | Time | No. of Variables | Degree n | Time |
|---|---|---|---|---|---|---|---|---|
| 1 | 10 | .55 | 2 | 2 | 1.11 | 3 | 1 | 2.36 |
| 1 | 20 | 1.35 | 2 | 4 | 3.83 | 3 | 2 | 8.93 |
| 1 | 30 | 2.23 | 2 | 6 | 7.56 | 3 | 3 | 23.7 |
| 1 | 40 | 3.60 | 2 | 8 | 13.6 | 3 | 4 | 49.5 |

## Polynomial Resultants

Times are given for PRES, for bivariate inputs in Table 3 and for trivariate inputs in Table 4.

### Table 3

The inputs to PRES are random bivariate polynomials A and B with 4-bit coefficients and with $\partial_i(A) = \partial_i(B) = n_i$ for $i = 1, 2$. The variable eliminated is $x_2$.

| | $n_2 = 1$ | $n_2 = 2$ | $n_2 = 3$ | $n_2 = 4$ | $n_2 = 5$ |
|---|---|---|---|---|---|
| $n_1 = 2$ | .05 | .10 | .35 | .58 | 1.29 |
| $n_1 = 4$ | .09 | .23 | .83 | 1.50 | 3.07 |
| $n_1 = 6$ | .15 | .39 | 1.48 | 2.72 | 5.54 |
| $n_1 = 8$ | .21 | .59 | 2.28 | 6.02 | 8.86 |
| $n_1 = 10$ | .29 | 1.50 | 3.61 | 9.05 | 13.08 |

Table 4

The inputs to PRES are random trivariate polynomials A and B with 4-bit coefficients, with $\partial_i(A) = \partial_i(B) = n$ for $i = 1,2$, and with $\partial_3(A) = \partial_3(B) = n_3$. The variable eliminated is $x_3$.

| | n = 1 | n = 2 | n = 3 | n = 4 | n = 5 |
|---|---|---|---|---|---|
| $n_3 = 2$ | .35 | 1.15 | 5.01 | 9.55 | 15.3 |
| $n_3 = 3$ | 1.53 | 5.92 | 14.10 | 25.8 | 45.2 |

## Polynomial Multiplication and Division

Computing times are given in Table 5 for multiplication, division and trial division of integral polynomials in one, two and three variables. For purposes of comparison, the computing times of the classical algorithms PPROD and PQ of the SAC-1 Polynomial System are listed alongside those of the modular algorithms PMPY, PDIV and PTDIV of the present system. Table 6 gives corresponding computing times for multiplication and division of polynomials over GF(p).

Table 5

The inputs to PPROD and PMPY are random polynomials A and B in one, two and three variables with $\partial_i(A) = \partial_i(B) = n$ for $i = 1,2,3$ and with single-precision (33-bit) and triple-precision (99-bit) coefficients. The inputs to PQ, PDIV and PTDIV are $A \cdot B$ and A, resulting in B as output.

| No. of Variables | Degree n | Coeff. Length | PPROD | PMPY | PQ | PDIV | PTDIV |
|---|---|---|---|---|---|---|---|
| 1 | 10 | 1 | .12 | .19 | .25 | .26 | .25 |
| 1 | 20 | 1 | .53 | .41 | .87 | .50 | .51 |
| 1 | 30 | 1 | 1.11 | .69 | 1.81 | .81 | .83 |
| 1 | 40 | 1 | 1.81 | 1.06 | 4.27 | 1.16 | 1.16 |
| 1 | 50 | 1 | 2.56 | 1.36 | 5.22 | 1.57 | 1.60 |
| 2 | 3 | 1 | .29 | .83 | .47 | .62 | .74 |
| 2 | 6 | 1 | 2.79 | 2.78 | 3.30 | 2.09 | 2.63 |
| 2 | 9 | 1 | 10.5 | 6.65 | 12.9 | 4.49 | 5.95 |
| 2 | 12 | 1 | 30.3 | 12.9 | 34.8 | 8.30 | 11.4 |
| 2 | 15 | 1 | 68.3 | 22.4 | 78.5 | 14.1 | 19.6 |
| 3 | 1 | 1 | .11 | .84 | .12 | .66 | .87 |
| 3 | 2 | 1 | 1.06 | 2.77 | 1.08 | 1.88 | 2.53 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 3 | 3 | 1 | 6.44 | 8.59 | 5.54 | 4.12 | 5.98 |
| 3 | 4 | 1 | 18.3 | 17.1 | 20.4 | 9.90 | 15.1 |
| 1 | 10 | 3 | .25 | .58 | .40 | .67 | .67 |
| 1 | 20 | 3 | .96 | 1.17 | 1.50 | 1.23 | 1.23 |
| 1 | 30 | 3 | 2.02 | 1.94 | 3.01 | 1.99 | 1.97 |
| 1 | 40 | 3 | 3.44 | 2.79 | 5.11 | 2.82 | 2.80 |
| 1 | 50 | 3 | 5.71 | 3.79 | 8.67 | 3.73 | 3.96 |
| 2 | 3 | 3 | .61 | 2.12 | .74 | 1.55 | 1.78 |
| 2 | 6 | 3 | 5.27 | 7.43 | 6.61 | 4.87 | 6.68 |
| 2 | 9 | 3 | 20.7 | 18.9 | 23.8 | 10.9 | 13.8 |
| 2 | 12 | 3 | 59.4 | 33.9 | 76.5 | 20.9 | 25.7 |
| 2 | 15 | 3 | 135.0 | 55.7 | 138.0 | 34.1 | 44.6 |
| 3 | 1 | 3 | .15 | 1.62 | .19 | 1.20 | 1.50 |
| 3 | 2 | 3 | 1.59 | 7.52 | 1.77 | 4.08 | 5.59 |
| 3 | 3 | 3 | 8.99 | 18.5 | 9.96 | 9.32 | 13.4 |
| 3 | 4 | 3 | 39.8 | 46.5 | 42.7 | 20.8 | 33.1 |

Table 6

The inputs to CPPROD and CPMPY are random polynomials A and B in one, two and three variables with $\partial_i(A) = \partial_i(B) = n$ for $i = 1,2,3$. The inputs to CPQ, CPDIV and CPTDIV are $A \cdot B$ and A, resulting in B as output.

| No. of Variables | Degree n | CPPROD | CPMPY | CPQ | CPDIV | CPTDIV |
|---|---|---|---|---|---|---|
| 1 | 10 | .011 | .012 | .011 | .012 | .013 |
| 1 | 20 | .033 | .035 | .033 | .034 | .039 |
| 1 | 30 | .066 | .080 | .066 | .072 | .073 |
| 1 | 40 | .113 | .135 | .113 | .121 | .125 |
| 1 | 50 | .173 | .216 | .194 | .185 | .213 |
| 2 | 3 | .04 | .13 | .04 | .06 | .11 |
| 2 | 6 | .26 | .58 | .26 | .22 | .43 |
| 2 | 9 | .96 | 1.59 | .96 | .54 | 1.15 |
| 2 | 12 | 2.60 | 3.66 | 2.59 | 1.09 | 2.33 |
| 2 | 15 | 5.98 | 6.17 | 6.22 | 1.97 | 3.94 |
| 2 | 18 | 11.6 | 10.1 | 11.6 | 3.08 | 6.44 |
| 2 | 21 | 20.6 | 13.8 | 19.9 | 4.72 | 9.48 |
| 2 | 24 | 33.4 | 21.6 | 32.5 | 6.59 | 13.8 |
| 3 | 2 | .15 | .59 | .18 | .21 | .43 |
| 3 | 4 | 2.15 | 3.77 | 2.22 | 1.03 | 2.22 |
| 3 | 6 | 13.9 | 12.9 | 14.4 | 3.43 | 7.44 |
| 3 | 8 | 55.2 | 34.8 | 56.4 | 8.34 | 19.0 |

## Chinese Remainder Algorithm

Consider the following complete algorithm for conversion of an integer from modular to radix representation.

$$B = CHRA(P,A)$$

P is a list $(p_1,\ldots,p_n)$ of distinct odd primes and A is a list $(a_1,\ldots,a_n)$ such that $0 \leq a_i < p_i$ for $1 \leq i \leq n$. B is the unique integer such that $B \equiv a_i \pmod{p_i}$ for $1 \leq i \leq n$ and $|B| < (\prod_{i=1}^{n} p_i)/2$.

Algorithm

(1) $Q \leftarrow PFA(1,0)$; $B \leftarrow 0$; $P' \leftarrow P$; ADV(p,P'); $A' \leftarrow A$; ADV(a,A'); $q \leftarrow 1$.

(2) $C \leftarrow CCRA(Q,B,p,a,q)$; ERLA(B); $B \leftarrow C$; if $P' = 0$, go to (4).

(3) IMFI(Q,p); ADV(p,P'); ADV(a,A'); $q \leftarrow CRECIP(p,CMOD(p,Q))$; go to (2).

(4) ERLA(Q); return.

If all the $p_i$ are about the same size, then the computing time of CHRA is a function of n. Table 7, following, gives the computing time of CHRA for n = 10, 20, 30, 40 and 50, when each $p_i$ is approximately $2^{31}$. The second column of the table shows how much of the CHRA time is attributable to the n applications of CCRA within CHRA. It can be deduced from this table that the computing time of CCRA(Q,B,p,a,q) is about .0004n seconds when Q is the product of n primes of this size.

Table 7

| No. of Primes r | CHRA | CCRA Applications |
|---|---|---|
| 10 | .047 | .021 |
| 20 | .162 | .070 |
| 30 | .368 | .153 |
| 40 | .671 | .256 |
| 50 | 1.098 | .384 |

## Evaluation and Interpolation

Let A be a polynomial over GF(p) in one, two or three variables, of degree n in each variable, and let $a_0,\ldots,a_n$ be $n+1$ distinct elements of GF(p). Table 8, following, shows, under the column labelled CPUEV, the time required to evaluate A at $x_1 = a_i$ using CPUEV, for $0 \leq i \leq n$. Under the column labelled CPINTI is given the time required to reconstruct the polynomial A from its values at $a_0,\ldots,a_n$ through the use of $n+1$ applications of CPINTI.

Table 8

| No. of Variables | Degree n | CPUEV Applications | CPINTI Applications |
|---|---|---|---|
| 1 | 20 | .02 | .13 |
| 1 | 40 | .08 | .45 |
| 1 | 60 | .17 | .93 |
| 1 | 80 | .30 | 1.55 |
| 1 | 100 | .47 | 2.37 |
| 2 | 10 | .07 | .28 |
| 2 | 20 | .42 | 1.37 |
| 2 | 30 | 1.38 | 3.77 |
| 2 | 40 | 3.05 | 8.18 |
| 3 | 3 | .02 | .12 |
| 3 | 6 | .14 | .67 |
| 3 | 9 | .52 | 2.14 |
| 3 | 12 | 1.40 | 5.18 |

Table 9

The time to compute n ! using IFACT.

| n | 50 | 100 | 150 | 200 | 250 |
|---|---|---|---|---|---|
| Time | .30 | 1.16 | 3.09 | 5.24 | 7.45 |

## 4. Fortran Program Listings

```
      INTEGER FUNCTION CCRA(QCAP,BCAP,P,A,Q)
      INTEGER A,B,BCAP,C,D,DCAP,P,Q,QCAP,T
      INTEGER BORROW,CDIF,CMOD,CPROD,PFA
1     B=CMOD(P,BCAP)
      C=CDIF(P,A,B)
      IF (C.NE.0) GO TO 2
      CCRA=BORROW(BCAP)
      RETURN
2     D=CPROD(P,C,Q)
      IF (D+D.GT.P) D=D-P
3     DCAP=PFA(D,0)
      T=IPROD(DCAP,QCAP)
      CALL ERLA(DCAP)
      CCRA=ISUM(T,BCAP)
      CALL ERLA(T)
      RETURN
      END

      SUBROUTINE CGCDCF(PS,A1P,A2P,BP,C1P,C2P)
      INTEGER AS1,AS1P,AS2,AS2P,A1,A1P,A1S,A2,A2P,A2S,B,BB,BBS,BP,BS,BSB
      INTEGER BSBS,BSM,BSP,C1,C1B,C1BS,C1P,C2,C2B,C2BS,C2P,D,DS,E,ES,HS
      INTEGER PS,RR,RS,S,TS
      INTEGER BORROW,CPDEG,CPDEGS,CPDEGT,CPEVAL,CPGCD1,CPLUCF,CPNV,CPONE
      INTEGER CPQ,CPUDIV,CPUEV,CPUPR,CSPROD,PFA,PFL
1     RR=1
      GO TO 10
2     RETURN
10    RS=CPNV(A1P)
      IF (RS.GT.1) GO TO 30
20    BP=CPGCD1(PS,A1P,A2P)
      C1P=CPQ(PS,A1P,BP)
      C2P=CPQ(PS,A2P,BP)
      GO TO 170
30    CALL CPUCPP(PS,A1P,AS1P,A1)
      CALL CPUCPP(PS,A2P,AS2P,A2)
      BSP=CPGCD1(PS,AS1P,AS2P)
40    AS1=CPLUCF(A1)
      AS2=CPLUCF(A2)
      BSB=CPGCD1(PS,AS1,AS2)
      CALL CPERAS(AS1)
      CALL CPERAS(AS2)
50    HS=CPDEG(BSB)
      LS1=CPDEG1(A1)+HS
      LS2=CPDEG1(A2)+HS
      LS=LS1
      IF (LS2.GT.LS1) LS=LS2
      KS=0
      ES=-1
      L=0
60    ES=ES+1
      IF (ES.NE.PS) GO TO 62
      PRINT 61
```

```
61    FORMAT(23H ALGORITHM CGCDCF FAILS)
      STOP
62    BSBS=CPEVAL(PS,BSB,ES)
      IF (BSBS.EQ.0) GO TO 60
70    A1S=CPUEV(PS,A1,ES)
      A2S=CPUEV(PS,A2,ES)
80    CALL STACK3(AS1P,AS2P,A1)
      CALL STACK3(A2,A1P,A2P)
      CALL STACK3(A1S,A2S,BB)
      CALL STACK3(BP,BSBS,BSP)
      CALL STACK3(C1B,C2B,C1P)
      CALL STACK3(C2P,D,ES)
      CALL STACK3(KS,L,LS)
      CALL STACK3(LS1,LS2,MS)
      CALL STACK3(NS1,NS2,BSB)
      CALL STACK2(RR,RS)
      RR=2
      A1P=A1S
      A2P=A2S
      GO TO 10
81    BS=BP
      C1BS=C1P
      C2BS=C2P
      CALL UNSTK2(RR,RS)
      CALL UNSTK3(NS1,NS2,BSB)
      CALL UNSTK3(LS1,LS2,MS)
      CALL UNSTK3(KS,L,LS)
      CALL UNSTK3(C2P,D,ES)
      CALL UNSTK3(C1B,C2B,C1P)
      CALL UNSTK3(BP,BSBS,BSP)
      CALL UNSTK3(A1S,A2S,BB)
      CALL UNSTK3(A2,A1P,A2P)
      CALL UNSTK3(AS1P,AS2P,A1)
90    CALL CPERAS(A1S)
      CALL CPERAS(A2S)
      IF (CPONE(BS).EQ.1) GO TO 91
      BBS=CSPROD(PS,BS,BSBS,0)
      CALL CPERAS(BS)
      GO TO 100
91    CALL CPERAS(BSB)
      CALL CPERAS(C1BS)
      CALL CPERAS(C2BS)
      CALL ERLA(L)
      IF (KS.EQ.0) GO TO 92
      CALL CPERAS(BB)
      CALL CPERAS(C1B)
      CALL CPERAS(C2B)
      CALL CPERAS(D)
92    B=PFA(0,PFL(BS,0))
      C1=A1
      C2=A2
      GO TO 160
```

```
100   LSB=CPDEGS(BBS)
      IF (L.NE.0) GO TO 101
      L=LSB
      GO TO 110
101   S=ICOMP(L,LSB)
      IF (S.NE.0) GO TO 102
      CALL ERLA(LSB)
      IF (KS.EQ.0) GO TO 110
      GO TO 120
102   IF (S.GT.0) GO TO 103
      CALL CPERAS(BBS)
      CALL CPERAS(C1BS)
      CALL CPERAS(C2BS)
      CALL ERLA(LSB)
      GO TO 60
103   CALL CPERAS(BB)
      CALL CPERAS(C1B)
      CALL CPERAS(C2B)
      CALL CPERAS(D)
      CALL ERLA(L)
      L=LSB
110   BB=0
      C1B=0
      C2B=0
      D=PFA(0,PFA(1,0))
      KS=0
      MS=0
      NS1=0
      NS2=0
120   E=BORROW(D)
      CALL CPINTI(PS,BB,ES,BBS,E,RS,TS)
      IF (TS.EQ.1) MS=KS
      CALL CPERAS(E)
      E=BORROW(D)
      CALL CPINTI(PS,C1B,ES,C1BS,E,RS,TS)
      IF (TS.EQ.1) NS1=KS
      CALL CPERAS(E)
      CALL CPINTI(PS,C2B,ES,C2BS,D,RS,TS)
      IF (TS.EQ.1) NS2=KS
      KS=KS+1
130   IF (MS+NS1.LE.LS1.AND.MS+NS2.LE.LS2) GO TO 131
      CALL CPERAS(BB)
      CALL CPERAS(C1B)
      CALL CPERAS(C2B)
      CALL CPERAS(D)
      KS=0
      GO TO 60
131   IF (KS.LE.LS) GO TO 60
140   CALL CPERAS(A1)
      CALL CPERAS(A2)
      CALL CPERAS(BSB)
      CALL CPERAS(D)
```

```
      CALL ERLA(L)
150   CALL CPUCPP(PS,BB,DS,B)
      CALL CPERAS(BB)
      CALL CPERAS(DS)
      BSM=CPLUCF(B)
      C1=CPUDIV(PS,C1B,BSM)
      CALL CPERAS(C1B)
      C2=CPUDIV(PS,C2B,BSM)
      CALL CPERAS(C2B)
      CALL CPERAS(BSM)
160   BP=CPUPR(PS,B,BSP)
      CALL CPERAS(B)
      DS=CPQ(PS,AS1P,BSP)
      CALL CPERAS(AS1P)
      C1P=CPUPR(PS,C1,DS)
      CALL CPERAS(C1)
      CALL CPERAS(DS)
      DS=CPQ(PS,AS2P,BSP)
      CALL CPERAS(AS2P)
      CALL CPERAS(BSP)
      C2P=CPUPR(PS,C2,DS)
      CALL CPERAS(C2)
      CALL CPERAS(DS)
170   GO TO (2,81),RR
      END

      INTEGER FUNCTION CPCOMP(X,Y)
      INTEGER A,B,C,D,R,T,U,X,Y
      INTEGER TYPE
      A=X
      B=Y
      R=1
      GO TO 10
 1    CPCOMP=T
      RETURN
C     RECURSIVE PROCEDURE T=CPCOMP(A,B)
 10   T=1
      IF (A.NE.0) GO TO 20
      IF (B.NE.0) GO TO (1,41),R
      T=0
      GO TO (1,41),R
 20   IF (B.EQ.0) GO TO (1,41),R
      CALL ADV(M,A)
      CALL ADV(N,B)
      IF (M.NE.N) GO TO (1,41),R
      U=TYPE(A)
 30   CALL ADV(C,A)
      CALL ADV(D,B)
      IF (U.NE.0) GO TO 40
      IF (C.NE.D) GO TO (1,41),R
      GO TO 50
```

```
C       BEGIN RECURSIVE PROCEDURE CALL T=CPCOMP(C,D)
 40     CALL STACK2(A,B)
        CALL STACK2(R,U)
        A=C
        B=D
        R=2
        GO TO 10
 41     CALL UNSTK2(R,U)
        CALL UNSTK2(A,B)
C       END RECURSIVE PROCEDURE CALL T=CPCOMP(C,D)
        IF (T.NE.0) GO TO (1,41),R
 50     IF (A.NE.0) GO TO 30
        T=0
        GO TO (1,41),R
C       END RECURSIVE PROCEDURE T=CPCOMP(A,B)
        END

        INTEGER FUNCTION CPCRA(QCAP,B1,P,A1,Q,L1)
        INTEGER A,ABAR,A1,B,BBAR,B1,C,E,F,G,P,Q,QCAP,R,S
        INTEGER BORROW,CCRA,FIRST,PFA,PFL,TAIL
  1     B=B1
        A=A1
        L=L1
        R=1
        GO TO 10
  2     CPCRA=C
        RETURN
 10     IF (L.NE.0) GO TO 20
        C=CCRA(QCAP,B,P,A,Q)
        GO TO 70
 20     C=0
        M=-1
        ABAR=A
        IF (ABAR.NE.0) CALL ADV(M,ABAR)
        N=-1
        BBAR=B
        IF (BBAR.EQ.0) GO TO 21
        BBAR=TAIL(BBAR)
        N=FIRST(TAIL(BBAR))
 21     MCAP=TAIL(L)
 30     IF (ABAR.EQ.0.AND.BBAR.EQ.0) GO TO 60
        E=0
        F=0
        S=M-N
        IF (S.LT.0) GO TO 31
        K=M
        M=M-1
        CALL ADV(E,ABAR)
 31     IF (S.GT.0) GO TO 32
        K=N
        N=-1
        CALL ADV(F,BBAR)
        BBAR=TAIL(BBAR)
        IF (BBAR.NE.0) N=FIRST(TAIL(BBAR))
```

```
32    IF (E.EQ.0.AND.F.EQ.0) GO TO 30
40    CALL STACK3(A,ABAR,B)
      CALL STACK3(BBAR,C,E)
      CALL STACK3(F,K,L)
      CALL STACK2(M,MCAP)
      CALL STACK2(N,R)
      B=F
      A=E
      L=MCAP
      R=2
      GO TO 10
41    G=C
      CALL UNSTK2(N,R)
      CALL UNSTK2(M,MCAP)
      CALL UNSTK3(F,K,L)
      CALL UNSTK3(BBAR,C,E)
      CALL UNSTK3(A,ABAR,B)
50    C=PFA(K,PFL(G,C))
      GO TO 30
60    IF (C.NE.0) C=PFL(BORROW(FIRST(L)),INV(C))
70    GO TO (2,41),R
      END

      INTEGER FUNCTION CPDEG(A)
      INTEGER A,FIRST
      CPDEG=0
      IF (A.NE.0) CPDEG=FIRST(A)
      RETURN
      END

      INTEGER FUNCTION CPDEGS(X)
      INTEGER A,B,H,K,L,M,N,R,T,U,V,X
      INTEGER FIRST,PFA,TAIL,TYPE
      A=X
      R=1
      GO TO 10
 1    CPDEGS=L
      RETURN
C     BEGIN RECURSIVE PROCEDURE L=CPDEGS(A)
10    N=FIRST(A)
      T=TAIL(A)
      IF (TYPE(T).NE.0) GO TO 20
      L=PFA(N,0)
      GO TO (1,21,31),R
20    CALL ADV(B,T)
C     BEGIN RECURSIVE PROCEDURE CALL L=CPDEGS(B)
      CALL STACK3(N,T,R)
      A=B
      R=2
      GO TO 10
```

```
 21   CALL UNSTK3(N,T,R)
C     END RECURSIVE PROCEDURE CALL L=CPDEGS(B)
 30   IF (T.EQ.0) GO TO 50
      CALL ADV(B,T)
      IF (B.EQ.0) GO TO 30
C     BEGIN RECURSIVE PROCEDURE CALL M=CPDEGS(B)
      CALL STACK2(N,T)
      CALL STACK2(L,R)
      A=B
      R=3
      GO TO 10
 31   M=L
      CALL UNSTK2(L,R)
      CALL UNSTK2(N,T)
C     END RECURSIVE PROCEDURE CALL M=CPDEGS(B)
      U=L
      V=M
 40   H=FIRST(U)
      CALL DECAP(K,V)
      IF (H.LT.K) CALL ALTER(K,U)
      U=TAIL(U)
      IF (U.NE.0) GO TO 40
      GO TO 30
 50   L=PFA(N,L)
      GO TO (1,21,31),R
C     END RECURSIVE PROCEDURE L=CPDEGS(A)
      END

      INTEGER FUNCTION CPDEG1(A)
      INTEGER A,CPDEGS,FIRST
  1   L=INV(CPDEGS(A))
      CPDEG1=FIRST(L)
      CALL ERLA(L)
      RETURN
      END

      INTEGER FUNCTION CPDIV(P,X,Y)
      INTEGER A,ASTAR,B,BSTAR,C,CSTAR,D,SL,SM,P,R,RR,T,X,Y
      INTEGER CPDEGS,CPNV,CPQREM,CPUEV,PFA
  1   A=X
      B=Y
      RR=1
      GO TO 10
  2   CPDIV=C
      RETURN
 10   C=0
      IF (A.EQ.0) GO TO 110
 20   R=CPNV(A)
      L=INV(CPDEGS(A))
      CALL DECAP(SL,L)
      CALL ERLA(L)
      M=INV(CPDEGS(B))
```

```
      CALL DECAP(SM,M)
      CALL ERLA(M)
      N=SL-SM
      IF (R.GT.1) GO TO 40
30    T=CPQREM(P,A,B)
      CALL DECAP(C,T)
      CALL DECAP(R,T)
      GO TO 110
40    I=-1
      D=PFA(0,PFA(1,0))
50    I=I+1
      BSTAR=CPUEV(P,B,I)
      IF (BSTAR.EQ.0) GO TO 50
60    ASTAR=CPUEV(P,A,I)
70    CALL STACK3(A,ASTAR,B)
      CALL STACK3(BSTAR,C,D)
      CALL STACK2(I,N)
      CALL STACK2(R,RR)
      A=ASTAR
      B=BSTAR
      RR=2
      GO TO 10
71    CSTAR=C
      CALL UNSTK2(R,RR)
      CALL UNSTK2(I,N)
      CALL UNSTK3(BSTAR,C,D)
      CALL UNSTK3(A,ASTAR,B)
      CALL CPERAS(ASTAR)
      CALL CPERAS(BSTAR)
80    CALL CPINTI(P,C,I,CSTAR,D,R,T)
90    IF (N.LE.0) GO TO 100
      N=N-1
      GO TO 50
100   CALL CPERAS(D)
      END

      SUBROUTINE CPINTI(P,A,B,C,D,R,T)
      INTEGER A,ASTAR,B,C,D,E,G,P,R,T,U,V,W
      INTEGER CDIF,CPDIF,CPEVAL,CPMPYI,CPPROD,CPSUM
      INTEGER CPUEV,CRECIP,CSPROD,PFA
10    ASTAR=CPUEV(P,A,B)
20    IF (R.NE.1) GO TO 21
      U=CDIF(P,C,ASTAR)
      GO TO 30
21    U=CPDIF(P,C,ASTAR)
      CALL CPERAS(C)
      CALL CPERAS(ASTAR)
30    T=1
      IF (U.NE.0) GO TO 40
      T=0
      GO TO 50
40    V=CPEVAL(P,D,B)
      V=CRECIP(P,V)
```

```
      E=CSPROD(P,D,V,0)
      W=CPMPYI(P,U,E,R)
      CALL CPERAS(E)
      IF (R.GT.1) CALL CPERAS(U)
      G=CPSUM(P,W,A)
      CALL CPERAS(A)
      CALL CPERAS(W)
      A=G
50    U=PFA(1,PFA(1,PFA(CDIF(P,0,B),0)))
      W=CPPROD(P,D,U)
      CALL CPERAS(D)
      CALL CPERAS(U)
      D=W
      RETURN
      END

      INTEGER FUNCTION CPLNCF(A)
      INTEGER A,B,T
      INTEGER FIRST,TAIL,TYPE
1     B=A
2     B=TAIL(B)
      T=TYPE(B)
      B=FIRST(B)
      IF (T.NE.0) GO TO 2
      CPLNCF=B
      RETURN
      END

      INTEGER FUNCTION CPLUCF(A)
      INTEGER A,B,C
      INTEGER BORROW,FIRST,TAIL,TYPE
10    B=A
20    C=FIRST(TAIL(B))
      IF (TYPE(TAIL(C)).NE.0) GO TO 21
      CPLUCF=BORROW(C)
      RETURN
21    B=C
      GO TO 20
      END

      INTEGER FUNCTION CPMON(P,A)
      INTEGER A,B,C,P
      INTEGER CPLNCF,CRECIP,CSPROD
1     B=CPLNCF(A)
      C=CRECIP(P,B)
      CPMON=CSPROD(P,A,C,0)
      RETURN
      END
```

```
      INTEGER FUNCTION CPMPY(P,X,Y)
      INTEGER A,ASTAR,B,BSTAR,C,CSTAR,D.CM,CN,P,R,S,T,X,Y
      INTEGER CPDEGS,CPUEV,CPNV,CPPROD,INV,PFA
1     A=X
      B=Y
      S=1
      GO TO 10
2     CPMPY=C
      RETURN
10    C=0
      IF (A.EQ.0.OR.B.EQ.0) GO TO 90
20    R=CPNV(A)
      IF (R.GT.1) GO TO 30
      C=CPPROD(P,A,B)
      GO TO 90
30    CM=INV(CPDEGS(A))
      CALL DECAP(M,CM)
      CALL ERLA(CM)
      CN=INV(CPDEGS(B))
      CALL DECAP(N,CN)
      CALL ERLA(CN)
      K=M+N
      D=PFA(0,PFA(1,0))
      I=0
40    ASTAR=CPUEV(P,A,I)
      BSTAR=CPUEV(P,B,I)
50    CALL STACK3(ASTAR,BSTAR,C)
      CALL STACK2(A,B)
      CALL STACK2(I,K)
      CALL STACK3(R,S,D)
      S=2
      A=ASTAR
      B=BSTAR
      GO TO 10
51    CSTAR=C
      CALL UNSTK3(R,S,D)
      CALL UNSTK2(I,K)
      CALL UNSTK2(A,B)
      CALL UNSTK3(ASTAR,BSTAR,C)
      CALL CPERAS(ASTAR)
      CALL CPERAS(BSTAR)
60    CALL CPINTI(P,C,I,CSTAR,D,R,T)
70    IF (I.EQ.K) GO TO 80
      I=I+1
      GO TO 40
80    CALL CPERAS(D)
90    GO TO (2,51),S
      END
```

```
      INTEGER FUNCTION CPMPYI(P,A1,B,R)
      INTEGER A,A1,B,C,D,E,P,R,RR,T
      INTEGER CSPROD,PFA,PFL
1     A=A1
      RR=1
      GO TO 10
2     CPMPYI=C
      RETURN
10    C=0
      IF (A.EQ.0.OR.B.EQ.0) GO TO 60
20    IF (R.NE.1) GO TO 30
      C=CSPROD(P,B,A,0)
      GO TO 60
30    T=A
      CALL ADV(N,T)
      C=PFA(N,C)
40    CALL ADV(D,T)
      CALL STACK3(A,C,D)
      CALL STACK3(R,RR,T)
      A=D
      R=R-1
      RR=2
      GO TO 10
41    E=C
      CALL UNSTK3(R,RR,T)
      CALL UNSTK3(A,C,D)
      C=PFL(E,C)
      IF (T.NE.0) GO TO 40
50    C=INV(C)
60    GO TO (2,41),RR
      END

      INTEGER FUNCTION CPNV(A)
      INTEGER A,B
      INTEGER FIRST,TAIL,TYPE
1     N=1
      B=A
2     L=TAIL(B)
      IF (TYPE(L).EQ.0) GO TO 4
3     B=FIRST(L)
      N=N+1
      GO TO 2
4     CPNV=N
      RETURN
      END

      INTEGER FUNCTION CPONE(X)
      INTEGER A,X,FIRST,TAIL,TYPE
      A=X
1     IF (A.EQ.0) GO TO 4
2     IF (FIRST(A).NE.0) GO TO 4
      A=TAIL(A)
```

```
      IF (TYPE(A).EQ.0) GO TO 3
      A=FIRST(A)
      GO TO 2
3     IF (FIRST(A).NE.1) GO TO 4
      CPONE=1
      RETURN
4     CPONE=0
      RETURN
      END

      INTEGER FUNCTION CPQ(P,A,B)
      INTEGER A,B,C,H,P
      INTEGER CPDEGS,CPQDB,PFA
 10   C=-1
      IF (B.NE.0) GO TO 11
      CPQ=C
      RETURN
 11   C=0
      IF (A.NE.0) GO TO 20
      CPQ=C
      RETURN
 20   MCAP=CPDEGS(A)
      NCAP=CPDEGS(B)
      LCAP=0
      H=0
 30   CALL DECAP(M,MCAP)
      CALL DECAP(N,NCAP)
      L=M-N
      LCAP=PFA(L,LCAP)
      IF (L.LT.0) H=L
      IF (MCAP.NE.0) GO TO 30
 40   LCAP=INV(LCAP)
      IF (H.GE.0) GO TO 50
      CPQ=-1
      CALL ERLA(LCAP)
      RETURN
 50   CPQ=CPQDB(P,A,B,LCAP)
      CALL ERLA(LCAP)
      RETURN
      END

      INTEGER FUNCTION CPQDB(P,A,B,L)
      INTEGER A,ABAR,B,BBAR,C,E,F,G,H,P,R,RR,S,T,V,W,Z
      INTEGER CINV,CPDEG,CPDIF,CPPROD,CPQREM,FIRST,PFA,PFL,TAIL
 1    RR=1
      GO TO 10
 2    CPQDB=C
      RETURN
 10   C=-1
      IF (B.EQ.0) GO TO 130
      C=0
      IF (A.EQ.0) GO TO 130
```

```
20    M=CPDEG(A)
      N=CPDEG(B)
      H=M-N
      IF (H.GE.0.AND.H.LE.FIRST(L)) GO TO 30
      C=-1
      GO TO 130
30    LSTAR=TAIL(L)
      IF (LSTAR.NE.0) GO TO 50
40    T=CPQREM(P,A,B)
      CALL DECAP(C,T)
      CALL DECAP(R,T)
      IF (R.EQ.0) GO TO 130
      CALL CPERAS(C)
      CALL CPERAS(R)
      C=-1
      GO TO 130
50    ABAR=TAIL(A)
      BBAR=TAIL(B)
      ABAR=INV(CINV(ABAR))
      CALL ADV(F,BBAR)
      C=PFA(H,0)
60    CALL DECAP(E,ABAR)
      CALL STACK3(A,B,C)
      CALL STACK3(ABAR,BBAR,H)
      CALL STACK2(E,F)
      CALL STACK3(L,LSTAR,RR)
      A=E
      B=F
      L=LSTAR
      RR=2
      GO TO 10
61    G=C
      CALL UNSTK3(L,LSTAR,RR)
      CALL UNSTK2(E,F)
      CALL UNSTK3(ABAR,BBAR,H)
      CALL UNSTK3(A,B,C)
      CALL CPERAS(E)
      IF (G.LT.0) GO TO 100
      C=PFL(G,C)
      IF (G.EQ.0) GO TO 80
      S=ABAR
      T=BBAR
70    IF (T.EQ.0) GO TO 80
      CALL ADV(W,T)
      Z=CPPROD(P,W,G)
      V=CPDIF(P,FIRST(S),Z)
      CALL CPERAS(FIRST(S))
      CALL ALTER(V,S)
      CALL CPERAS(Z)
      S=TAIL(S)
      GO TO 70
80    H=H-1
```

```
      IF (H.GE.0) GO TO 60
90    C=INV(C)
100   IF (ABAR.EQ.0) GO TO 130
      IF (FIRST(ABAR).NE.0) GO TO 110
      CALL DECAP(W,ABAR)
      GO TO 100
110   CALL DECAP(W,ABAR)
      CALL CPERAS(W)
      IF (ABAR.NE.0) GO TO 110
120   CALL CPERAS(C)
      C=-1
130   GO TO (2,61),RR
      END

      INTEGER FUNCTION CPRES(P,X,Y)
      INTEGER A,ASTAR,B,BSTAR,C,CSTAR,D,P,R,RR,T,X,Y
      INTEGER CPDEG,CPDEGS,CPNV,CPRES1,CPUEV,PFA,TAIL,TYPE
1     A=X
      B=Y
      RR=1
      GO TO 10
2     CPRES=C
      RETURN
10    IF (TYPE(TAIL(A)).NE.0) GO TO 20
      C=CPRES1(P,A,B)
      GO TO 110
20    MCAP=CPDEGS(A)
      CALL DECAP(M,MCAP)
      MCAP=INV(MCAP)
      CALL DECAP(M1,MCAP)
      CALL ERLA(MCAP)
30    NCAP=CPDEGS(B)
      CALL DECAP(N,NCAP)
      NCAP=INV(NCAP)
      CALL DECAP(N1,NCAP)
      CALL ERLA(NCAP)
40    K=M*N1+N*M1
      R=CPNV(A)-1
      C=0
      D=PFA(0,PFA(1,0))
      I=-1
50    I=I+1
      IF (I.NE.P) GO TO 60
      PRINT 51
51    FORMAT(22H ALGORITHM CPRES FAILS)
      STOP
60    ASTAR=CPUEV(P,A,I)
      IF (CPDEG(ASTAR).GE.M) GO TO 70
      CALL CPERAS(ASTAR)
      GO TO 50
70    BSTAR=CPUEV(P,B,I)
      IF (CPDEG(BSTAR).GE.N) GO TO 80
```

```
      CALL CPERAS(ASTAR)
      CALL CPERAS(BSTAR)
      GO TO 50
80    CALL STACK3(A,ASTAR,B)
      CALL STACK3(BSTAR,C,D)
      CALL STACK3(I,K,M)
      CALL STACK3(N,R,RR)
      A=ASTAR
      B=BSTAR
      RR=2
      GO TO 10
81    CSTAR=C
      CALL UNSTK3(N,R,RR)
      CALL UNSTK3(I,K,M)
      CALL UNSTK3(BSTAR,C,D)
      CALL UNSTK3(A,ASTAR,B)
      CALL CPERAS(ASTAR)
      CALL CPERAS(BSTAR)
90    CALL CPINTI(P,C,I,CSTAR,D,R,T)
      IF (CPDEG(D).LE.K) GO TO 50
100   CALL CPERAS(D)
110   GO TO (2,81),RR
      END

      INTEGER FUNCTION CPRESI(P,A,B)
      INTEGER A,A1,A2,A3,B,C,C2,I,N,N1,N2,N3,NU,P
      INTEGER BORROW,FIRST,TAIL,CPREM,CPROD
10    A1=BORROW(A)
      A2=BORROW(B)
      NU=0
      C=1
20    A3=CPREM(P,A1,A2)
      IF (A3.NE.0) GO TO 30
      C=0
      GO TO 90
30    N1=FIRST(A1)
      N2=FIRST(A2)
      N3=FIRST(A3)
40    M1=MOD(N1,2)
      M2=MOD(N2,2)
      NU=MOD(NU+M1*M2,2)
50    N=N1-N3
      C2=FIRST(TAIL(A2))
60    DO 61 I=1,N
61    C=CPROD(P,C,C2)
70    CALL CPERAS(A1)
      A1=A2
      A2=A3
      IF (N3.GT.0) GO TO 20
80    C2=FIRST(TAIL(A2))
      DO 81 I=1,N2
81    C=CPROD(P,C,C2)
```

```
      IF (NU.NE.0) C=P*C
90    CALL CPERAS(A1)
      CALL CPERAS(A2)
      CPRES1=C
      RETURN
      END

      INTEGER FUNCTION CPTDIV(P,A,B)
      INTEGER A,ASTAR,B,BSTAR,C,CSTAR,D,P,R,RCAP,RR,T,U
      INTEGER CPDEG,CPDEGS,CPNV,CPQREM,CPUEV,PFA
1     RR=1
      GO TO 10
2     CPTDIV=C
      RETURN
10    C=-1
      IF (B.EQ.0) GO TO 130
      C=0
      IF (A.EQ.0) GO TO 130
20    T=CPDEGS(A)
      U=CPDEGS(B)
30    CALL DECAP(L,T)
      CALL DECAP(M,U)
      N=L-M
      IF (N.LT.0) C=-1
      IF (T.NE.0) GO TO 30
40    IF (C.EQ.-1) GO TO 130
      R=CPNV(A)
      IF (R.GT.1) GO TO 60
50    T=CPQREM(P,A,B)
      CALL DECAP(C,T)
      CALL DECAP(RCAP,T)
      IF (RCAP.EQ.0) GO TO 130
      CALL CPERAS(C)
      CALL CPERAS(RCAP)
      C=-1
      GO TO 130
60    I=-1
      J=0
      D=PFA(0,PFA(1,0))
70    I=I+1
      IF (I.EQ.P) STOP
      ASTAR=CPUEV(P,A,I)
      BSTAR=CPUEV(P,B,I)
      IF (ASTAR.EQ.0.AND.BSTAR.EQ.0) GO TO 70
80    CALL STACK3(A,ASTAR,B)
      CALL STACK3(BSTAR,C,D)
      CALL STACK3(I,J,L)
      CALL STACK3(N,R,RR)
      A=ASTAR
      B=BSTAR
      RR=2
      GO TO 10
```

```
81    CSTAR=C
      CALL UNSTK3(N,R,RR)
      CALL UNSTK3(I,J,L)
      CALL UNSTK3(BSTAR,C,D)
      CALL UNSTK3(A,ASTAR,B)
      CALL CPERAS(ASTAR)
      CALL CPERAS(BSTAR)
      IF (CSTAR.EQ.-1) GO TO 110
90    CALL CPINTI(P,C,I,CSTAR,D,R,T)
      J=J+1
100   IF (CPDEG(C).GT.N) GO TO 110
      IF (J.LE.L) GO TO 70
      GO TO 120
110   CALL CPERAS(C)
      C=-1
120   CALL CPERAS(D)
130   GO TO (2,81),RR
      END

      INTEGER FUNCTION CPUCNT(P,A)
      INTEGER BORROW,CPGCD1,CPONE,FIRST,PFA,TAIL,TYPE
      INTEGER A,B,C,D,P,S
10    B=0
      C=A
      S=0
20    L=TAIL(C)
      IF (TYPE(L).EQ.0) GO TO 30
      S=PFA(TAIL(L),S)
      C=FIRST(L)
      GO TO 20
30    IF (B.NE.0) GO TO 31
      B=BORROW(C)
      GO TO 40
31    D=CPGCD1(P,B,C)
      CALL CPERAS(B)
      B=D
      IF (CPONE(B).NE.1) GO TO 40
      CALL ERLA(S)
      GO TO 50
40    IF (S.EQ.0) GO TO 50
      CALL DECAP(L,S)
      IF (L.EQ.0) GO TO 40
      CALL ADV(C,L)
      S=PFA(L,S)
      GO TO 20
50    CPUCNT=B
      RETURN
      END
```

```
      SUBROUTINE CPUCPP(P,A,C,ABAR)
      INTEGER A,ABAR,C,P,T
      INTEGER BORROW,CPONE,CPUDIV,CPUCNT
 10   C=CPUCNT(P,A)
      T=CPONE(C)
      IF (T.EQ.1) GO TO 11
      ABAR=CPUDIV(P,A,C)
      RETURN
 11   ABAR=BORROW(A)
      RETURN
      END

      INTEGER FUNCTION CPUDIV(P,A1,B1)
      INTEGER A,A1,B,B1,C,D,E,P,R
      INTEGER CPQ,FIRST,PFA,PFL,TAIL,TYPE
      A=A1
      B=B1
      R=1
      GO TO 10
 1    CPUDIV=C
      RETURN
 10   IF (TYPE(TAIL(A)).NE.0) GO TO 11
      C=CPQ(P,A,B)
      GO TO 40
 11   N=FIRST(A)
      C=PFA(N,0)
      L=TAIL(A)
 20   CALL ADV(D,L)
C ***  RECURSIVE CALL E=CPUDIV(P,D,B)
      CALL STACK2(C,A)
      CALL STACK2(R,L)
      A=D
      R=2
      GO TO 10
C ***  RETURN FROM RECURSIVE CALL
 21   E=C
      CALL UNSTK2(R,L)
      CALL UNSTK2(C,A)
      C=PFL(E,C)
      IF (L.NE.0) GO TO 20
 30   C=INV(C)
 40   GO TO (1,21),R
      END

      INTEGER FUNCTION CPUEV(P,A1,E)
      INTEGER A,A1,ABAR,B,C,D,E,P,R,RR
      INTEGER CPDEG,CPEVAL,CPNV,PFA,PFL,TAIL
      A=A1
      RR=1
      GO TO 10
 1    CPUEV=B
      RETURN
```

```
 10     B=0
        IF (A.EQ.0) GO TO 50
        R=CPNV(A)
        IF (R.GT.1) GO TO 11
        B=CPEVAL(P,A,E)
        GO TO 50
 11     N=CPDEG(A)
        ABAR=TAIL(A)
 20     CALL ADV(C,ABAR)
        IF (R.GT.2) GO TO 30
        D=CPEVAL(P,C,E)
        IF (B.EQ.0.AND.D.EQ.0) GO TO 40
        IF (B.EQ.0) B=PFA(N,B)
        B=PFA(D,B)
        GO TO 40
C ***   RECURSIVE CALL D=CPUEV(P,C,E)
 30     CALL STACK3(ABAR,B,N)
        CALL STACK2(R,RR)
        A=C
        RR=2
        GO TO 10
C ***   RETURN FROM RECURSIVE CALL
 31     D=B
        CALL UNSTK2(R,RR)
        CALL UNSTK3(ABAR,B,N)
        IF (B.EQ.0.AND.D.EQ.0) GO TO 40
        IF (B.EQ.0) B=PFA(N,B)
        B=PFL(D,B)
 40     N=N-1
        IF (ABAR.NE.0) GO TO 20
        B=INV(B)
 50     GO TO (1,31),RR
        END

        INTEGER FUNCTION CPUPR(P,A1,B1)
        INTEGER A,A1,B,B1,C,D,E,P,R
        INTEGER CPPROD,FIRST,PFA,PFL,TAIL,TYPE
        A=A1
        B=B1
        R=1
        GO TO 10
 1      CPUPR=C
        RETURN
 10     IF (TYPE(TAIL(A)).NE.0) GO TO 11
        C=CPPROD(P,A,B)
        GO TO 40
 11     N=FIRST(A)
        C=PFA(N,0)
        L=TAIL(A)
 20     CALL ADV(D,L)
C ***   RECURSIVE CALL E=CPUPR(P,D,B)
        CALL STACK2(A,L)
```

```
      CALL STACK2(C,R)
      A=D
      R=2
      GO TO 10
C *** RETURN FROM RECURSIVE CALL
 21   E=C
      CALL UNSTK2(C,R)
      CALL UNSTK2(A,L)
      C=PFL(E,C)
      IF (L.NE.0) GO TO 20
 30   C=INV(C)
 40   GO TO (1,21),R
      END

      INTEGER FUNCTION IFACT(N)
      INTEGER PFA
 1    L=PFA(1,0)
      IF (N.LT.2) GO TO 3
      I=2
 2    CALL IMFI(L,I)
      I=I+1
      IF (I.LE.N) GO TO 2
 3    IFACT=L
      RETURN
      END

      SUBROUTINE IMFI(A,B)
      INTEGER A,B,BCAP,C,PFA
 1    IF (B.NE.0) GO TO 3
 2    CALL ERLA(A)
      A=0
      RETURN
 3    BCAP=PFA(B,0)
      C=IPROD(A,BCAP)
      CALL ERLA(A)
      CALL ERLA(BCAP)
      A=C
      RETURN
      END

      INTEGER FUNCTION PCOMP(X,Y)
      INTEGER A,B,E,F,R,T,X,Y
      INTEGER TAIL,TYPE
      A=X
      B=Y
      R=1
      GO TO 1
 20   PCOMP=T
      RETURN
C     BEGIN RECURSIVE PROCEDURE T=PCOMP(A,B)
  1   IF (A.NE.0) GO TO 2
      IF (B.NE.0) GO TO 10
```

```
      GO TO 9
    2 IF (B.EQ.0) GO TO 10
      IF (TYPE(A).NE.0) GO TO 5
    3 CALL ADV(E,A)
      CALL ADV(F,B)
      IF (E.NE.F) GO TO 10
      IF (A.EQ.0) GO TO 4
      IF (B.NE.0) GO TO 3
      GO TO 10
    4 IF (B.EQ.0) GO TO 9
      GO TO 10
    5 A=TAIL(A)
      B=TAIL(B)
    6 CALL ADV(E,A)
      CALL ADV(F,B)
C     BEGIN RECURSIVE PROCEDURE CALL T=PCOMP(E,F)
      CALL STACK3(A,B,R)
      A=E
      B=F
      R=2
      GO TO 1
   61 CALL UNSTK3(A,B,R)
C     END RECURSIVE PROCEDURE CALL T=PCOMP(E,F)
    7 IF (T.EQ.1) GO TO 11
      CALL ADV(E,A)
      CALL ADV(F,B)
      IF (E.NE.F) GO TO 10
      IF (A.EQ.0) GO TO 8
      IF (B.NE.0) GO TO 6
      GO TO 10
    8 IF (B.NE.0) GO TO 10
    9 T=0
      GO TO 11
   10 T=1
   11 GO TO (20,61),R
C     END RECURSIVE PROCEDURE T=PCOMP(A,B)
      END

      INTEGER FUNCTION PCONT(A)
      INTEGER A,B,C,D,T
      INTEGER PGCD,PONE,TAIL
   10 B=0
      IF (A.NE.0) GO TO 11
      PCONT=0
      RETURN
   11 T=TAIL(A)
   20 CALL ADV(C,T)
      D=PGCD(B,C)
      CALL PERASE(B)
      B=D
      IF (PONE(B).NE.1) GO TO 21
      PCONT=B
```

```
      RETURN
 21   T=TAIL(T)
      IF (T.NE.0) GO TO 20
      PCONT=8
      RETURN
      END

      INTEGER FUNCTION PDEGS(X)
      INTEGER A,C,R,S,T,V,X
      INTEGER FIRST,PDEG,PFA,TAIL,TYPE
      A=X
      R=1
      GO TO 10
C ***  RETURN TO CALLING PROGRAM
 1    PDEGS=L
      RETURN
C ***  RECURSIVE PROCEDURE
 10   L=0
      IF (A.EQ.0) GO TO 90
      N=PDEG(A)
      T=TAIL(A)
      M=FIRST(T)
 20   IF (TYPE(M).NE.0) GO TO 40
 30   L=PFA(N,0)
      GO TO 90
 40   L=PFA(0,L)
      M=FIRST(TAIL(M))
      IF (TYPE(M).NE.0) GO TO 40
 50   C=FIRST(T)
C ***  RECURSIVE CALL K=PDEGS(C)
      CALL STACK2(R,L)
      CALL STACK2(N,T)
      A=C
      R=2
      GO TO 10
C ***  RETURN FROM RECURSIVE CALL
 51   K=L
      CALL UNSTK2(N,T)
      CALL UNSTK2(R,L)
      V=L
 60   CALL DECAP(S,K)
      IF (S.GT.FIRST(V)) CALL ALTER(S,V)
      V=TAIL(V)
      IF (V.NE.0) GO TO 60
 70   T=TAIL(TAIL(T))
      IF (T.NE.0) GO TO 50
 80   L=PFA(N,L)
 90   GO TO (1,51),R
      END
```

```
      INTEGER FUNCTION PDIV(A,B)
      COMMON /TR4/PRIME,PEXP
      INTEGER A,ASTAR,B,BSTAR,C,CSTAR,C1,F,G,H,P,PEXP,PRIME,Q,QCAP,S,T
      INTEGER CMOD,CPCRA,CPDIV,CPMOD,CRECIP,PCOMP,PFA,PNORMF,PVLIST
10    C=0
      IF (A.NE.0) GO TO 20
      PDIV=C
      RETURN
20    QCAP=PNORMF(A)
      CALL ELPOF2(QCAP,T,H)
      CALL ERLA(QCAP)
      QCAP=PNORMF(B)
      CALL ELPOF2(QCAP,T,K)
      CALL ERLA(QCAP)
      F=PEXP
      M=0
      N=0
      QCAP=PFA(1,0)
      I=PRIME
      L=INV(PVLIST(A))
30    IF (I.EQ.0) STOP
      CALL ADV(P,I)
      BSTAR=CPMOD(P,B)
      IF (BSTAR.EQ.0) GO TO 30
40    ASTAR=CPMOD(P,A)
      CSTAR=CPDIV(P,ASTAR,BSTAR)
      CALL CPERAS(ASTAR)
      CALL CPERAS(BSTAR)
50    Q=CRECIP(P,CMOD(P,QCAP))
      C1=CPCRA(QCAP,C,P,CSTAR,Q,L)
      S=PCOMP(C,C1)
      CALL PERASE(C)
      CALL CPERAS(CSTAR)
      C=C1
      CALL IMFI(QCAP,P)
      M=M+1
      IF (S.EQ.0) GO TO 51
      N=M
      GO TO 30
51    G=M*F
      IF (G.LE.H.OR.G.LT.K+N*(F+1)) GO TO 30
60    CALL ERASE(L)
      CALL ERLA(QCAP)
      PDIV=C
      RETURN
      END

      SUBROUTINE PGCDCF(A1P,A2P,BP,C1P,C2P)
      INTEGER AS1,AS1P,AS2,AS2P,A1,A1P,A1S,A2,A2P,A2S,B,BB,BBS,BP,BS
      INTEGER BSB,BSBS,BSM,BSP,C1,C1B,C1BS,C1P,C2,C2B,C2BS,C2P,DS,HS
      INTEGER P,PEXP,PRIME,PS,Q,QS,S,T,TS,TS1,TS2,V
      INTEGER BORROW,CMOD,CPCRA,CPDEGS,CPMOD,CPONE,CRECIP,CSPROD,PFA
```

```
      INTEGER PICONT,PID,PIP,PLNCF,PNORMF,PORDER,PVLIST,TYPE
      COMMON /TR4/PRIME,PEXP
10    IF (TYPE(A1P).NE.0) GO TO 20
      BP=IGCD(A1P,A2P)
      C1P=IQ(A1P,BP)
      C2P=IQ(A2P,BP)
      RETURN
20    AS1P=PICONT(A1P)
      A1=PID(A1P,AS1P)
      AS2P=PICONT(A2P)
      A2=PID(A2P,AS2P)
      BSP=IGCD(AS1P,AS2P)
30    AS1=PLNCF(A1)
      AS2=PLNCF(A2)
      BSB=IGCD(AS1,AS2)
      CALL ERLA(AS1)
      CALL ERLA(AS2)
40    LS1=PNORMF(A1)
      LS2=PNORMF(A2)
      S=ICOMP(LS1,LS2)
      IF (S.LE.0) GO TO 41
      LS=LS1
      CALL ERLA(LS2)
      GO TO 42
41    LS=LS2
      CALL ERLA(LS1)
42    TS=BORROW(BSB)
      CALL IMFI(TS,2)
      HS=IPROD(LS,TS)
      CALL ERLA(LS)
      CALL ERLA(TS)
      P=PRIME
      L=0
      V=INV(PVLIST(A1))
50    IF (P.NE.0) GO TO 52
      PRINT 51
      STOP
51    FORMAT(23H ALGORITHM PGCDCF FAILS)
52    CALL ADV(PS,P)
      BSBS=CMOD(PS,BSB)
      IF (BSBS.EQ.0) GO TO 50
60    A1S=CPMOD(PS,A1)
      A2S=CPMOD(PS,A2)
70    CALL CGCDCF(PS,A1S,A2S,BS,C1BS,C2BS)
80    CALL CPERAS(A1S)
      CALL CPERAS(A2S)
      IF (CPONE(BS).NE.1) GO TO 100
90    IF (L.EQ.0) GO TO 91
      CALL ERLA(L)
      CALL PERASE(BB)
      CALL PERASE(C1B)
      CALL PERASE(C2B)
```

```
      CALL ERLA(Q)
91    CALL CPERAS(BS)
      CALL CPERAS(C1BS)
      CALL CPERAS(C2BS)
      CALL ERLA(BSB)
      CALL ERLA(HS)
      V=INV(V)
      TS=PFA(1,0)
      B=PORDER(TS,V)
      CALL ERLA(TS)
      CALL ERASE(V)
      C1=A1
      C2=A2
      GO TO 170
100   BBS=CSPROD(PS,BS,BSBS,0)
      CALL CPERAS(BS)
110   LST=CPDEGS(BBS)
      IF (L.NE.0) GO TO 111
      L=LST
      GO TO 120
111   S=ICOMP(L,LST)
      IF (S.NE.0) GO TO 112
      CALL ERLA(LST)
      GO TO 130
112   IF (S.GT.0) GO TO 113
      CALL CPERAS(BBS)
      CALL CPERAS(C1BS)
      CALL CPERAS(C2BS)
      CALL ERLA(LST)
      GO TO 150
113   CALL PERASE(BB)
      CALL PERASE(C1B)
      CALL PERASE(C2B)
      CALL ERLA(Q)
      CALL ERLA(L)
120   BB=0
      C1B=0
      C2B=0
      Q=PFA(1,0)
130   QS=CRECIP(PS,CMOD(PS,Q))
      T=CPCRA(Q,BB,PS,BBS,QS,V)
      CALL PERASE(BB)
      CALL CPERAS(BBS)
      BB=T
      T=CPCRA(Q,C1B,PS,C1BS,QS,V)
      CALL PERASE(C1B)
      CALL CPERAS(C1BS)
      C1B=T
      T=CPCRA(Q,C2B,PS,C2BS,QS,V)
      CALL PERASE(C2B)
      CALL CPERAS(C2BS)
      C2B=T
```

```
      CALL IMFI(Q,PS)
140   S=ICOMP(Q,HS)
      IF (S.LE.0) GO TO 50
      MS=PNORMF(BB)
      CALL IMFI(MS,2)
      NS1=PNORMF(C1B)
      NS2=PNORMF(C2B)
      MS1=IPROD(MS,NS1)
      MS2=IPROD(MS,NS2)
      TS1=ICOMP(Q,MS1)
      TS2=ICOMP(Q,MS2)
      CALL ERLA(MS)
      CALL ERLA(NS1)
      CALL ERLA(NS2)
      CALL ERLA(MS1)
      CALL ERLA(MS2)
      IF (TS1.LE.0.OR.TS2.LE.0) GO TO 50
150   CALL PERASE(A1)
      CALL PERASE(A2)
      CALL ERLA(BSB)
      CALL ERLA(Q)
      CALL ERLA(L)
      CALL ERASE(V)
      CALL ERLA(HS)
160   DS=PICONT(BB)
      B=PID(BB,DS)
      CALL PERASE(BB)
      CALL ERLA(DS)
      BSM=PLNCF(B)
      C1=PID(C1B,BSM)
      CALL PERASE(C1B)
      C2=PID(C2B,BSM)
      CALL PERASE(C2B)
      CALL ERLA(BSM)
170   BP=PIP(B,BSP)
      CALL PERASE(B)
      DS=IQ(AS1P,BSP)
      CALL ERLA(AS1P)
      C1P=PIP(C1,DS)
      CALL PERASE(C1)
      CALL ERLA(DS)
      DS=IQ(AS2P,BSP)
      CALL ERLA(AS2P)
      CALL ERLA(BSP)
      C2P=PIP(C2,DS)
      CALL PERASE(C2)
      CALL ERLA(DS)
      RETURN
      END
```

```
      INTEGER FUNCTION PGCD(A,B)
      INTEGER A,ABAR,B,BBAR,C,PABS,TYPE
10    IF (A.NE.0) GO TO 11
      PGCD=PABS(B)
      RETURN
11    IF (B.NE.0) GO TO 20
      PGCD=PABS(A)
      RETURN
20    IF (TYPE(A).NE.0) GO TO 30
      PGCD=IGCD(A,B)
      RETURN
30    CALL PGCDCF(A,B,C,ABAR,BBAR)
      PGCD=C
      CALL PERASE(ABAR)
      CALL PERASE(BBAR)
      RETURN
      END

      INTEGER FUNCTION PICONT(A)
      INTEGER A,B,C,S,T
      INTEGER FIRST,PFA,TAIL,TYPE
10    B=A
      I=0
      S=0
20    B=TAIL(B)
      C=FIRST(B)
      IF (TYPE(C).EQ.0) GO TO 30
      S=PFA(TAIL(TAIL(B)),S)
      B=C
      GO TO 20
30    T=IGCD(I,C)
      CALL ERLA(I)
      I=T
      IF (FIRST(I).NE.1) GO TO 31
      IF (TAIL(I).NE.0) GO TO 31
      CALL ERLA(S)
      PICONT=I
      RETURN
31    B=TAIL(TAIL(B))
      IF (B.EQ.0) GO TO 40
      C=FIRST(B)
      GO TO 30
40    IF (S.NE.0) GO TO 41
      PICONT=I
      RETURN
41    CALL DECAP(B,S)
      IF (B.EQ.0) GO TO 40
      T=FIRST(B)
      S=PFA(TAIL(TAIL(B)),S)
      B=T
      GO TO 20
      END
```

```
      INTEGER FUNCTION PID(A1,I1)
      INTEGER A,A1,B,C,D,E,R,T
      INTEGER BORROW,FIRST,PFA,PFL,PVBL,TAIL,TYPE
      A=A1
      I=I1
      R=1
      GO TO 10
1     PID=B
      RETURN
10    B=0
      IF (A.EQ.0) GO TO 40
      IF (FIRST(I).NE.1) GO TO 11
      IF (TAIL(I).NE.0) GO TO 11
      B=BORROW(A)
      GO TO 40
11    B=PFL(PVBL(A),0)
      L=TAIL(A)
20    CALL ADV(C,L)
      T=TYPE(C)
      IF (T.EQ.0) D=IQ(C,I)
      IF (T.NE.1) GO TO 22
      CALL STACK3(L,B,R)
      A=C
      R=2
      GO TO 10
21    D=B
      CALL UNSTK3(L,B,R)
22    CALL ADV(E,L)
      B=PFA(E,PFL(D,B))
      IF (L.NE.0) GO TO 20
30    B=INV(B)
40    GO TO (1,21),R
      END

      INTEGER FUNCTION PLNCF(A)
      INTEGER A,B,BORROW,FIRST,TYPE,TAIL
10    IF (A.NE.0) GO TO 11
      PLNCF=0
      RETURN
11    B=A
      GO TO 30
20    B=FIRST(TAIL(B))
30    IF (TYPE(B).NE.0) GO TO 20
      PLNCF=BORROW(B)
      RETURN
      END

      INTEGER FUNCTION PMNORM(A)
      INTEGER A,C,S,T,TAIL,ICOMP,PNORMF
1     N=0
      T=TAIL(A)
2     CALL ADV(C,T)
```

```
      M=PNORMF(C)
      S=ICOMP(M,N)
      IF (S.GT.0) GO TO 3
      CALL ERLA(M)
      GO TO 4
3     CALL ERLA(N)
      N=M
4     T=TAIL(T)
      IF (T.NE.0) GO TO 2
      PMNORM=N
      RETURN
      END

      INTEGER FUNCTION PMPY(A,B)
      INTEGER A,ASTAR,B,BSTAR,C,CSTAR,C1,D,F,F,P,PEXP,PRIME,Q,QCAP
      INTEGER CMOD,CPCRA,CPMOD,CPMPY,CRECIP,PFA,PNORMF,PVLIST
      COMMON /TR4/ PRIME,PEXP
1     C=0
      PMPY=C
      IF (A.EQ.0.OR.B.EQ.0) RETURN
2     D=PNORMF(A)
      E=PNORMF(B)
      F=IPROD(D,E)
      CALL ERLA(D)
      CALL ERLA(E)
      CALL IMFI(F,2)
3     I=PRIME
      L=PVLIST(A)
      QCAP=PFA(I,0)
      L=INV(L)
4     IF (I.EQ.0) STOP
      CALL ADV(P,I)
      ASTAR=CPMOD(P,A)
      BSTAR=CPMOD(P,B)
5     CSTAR=CPMPY(P,ASTAR,BSTAR)
      CALL CPERAS(ASTAR)
      CALL CPERAS(BSTAR)
6     Q=CRECIP(P,CMOD(P,QCAP))
      C1=CPCRA(QCAP,C,P,CSTAR,Q,L)
      CALL CPERAS(CSTAR)
      CALL PERASE(C)
      C=C1
7     CALL IMFI(QCAP,P)
      IF (ICOMP(QCAP,F).LT.0) GO TO 4
8     CALL ERASE(L)
      CALL ERLA(F)
      CALL ERLA(QCAP)
      PMPY=C
      RETURN
      END
```

```
      INTEGER FUNCTION PRES(A,B)
      INTEGER A,ASTAR,B,BSTAR,C,CSTAR,D,E,F,P,PRIME,Q,QS,S,T,U,V
      INTEGER PFA,FIRST,TAIL,ICOMP,IFACT,IPROD,ISUM
      INTEGER CMOD,CPCRA,CPDEG,CPMOD,CPRES,CRECIP
      INTEGER PDEG,PMNORM,PPOWER,PVLIST
      COMMON /TR4/PRIME,PEXP
10    D=PMNORM(A)
      E=PMNORM(B)
      M=PDEG(A)
      N=PDEG(B)
      T=PPOWER(D,N)
      U=PPOWER(E,M)
      V=IPROD(T,U)
      CALL ERLA(T)
      CALL ERLA(U)
      T=ISUM(V,V)
      CALL ERLA(V)
      U=IFACT(M*N)
      F=IPROD(T,U)
      CALL ERLA(T)
      CALL ERLA(U)
      CALL ERLA(D)
      CALL ERLA(E)
20    I=PRIME
      Q=PFA(I,0)
      C=0
      L=PVLIST(FIRST(TAIL(A)))
      L=INV(L)
30    IF (I.NE.0) GO TO 32
      PRINT 31
31    FORMAT (21H ALGORITHM PRES FAILS)
      STOP
32    CALL ADV(P,I)
40    ASTAR=CPMOD(P,A)
      IF (CPDEG(ASTAR).EQ.M) GO TO 50
      CALL CPERAS(ASTAR)
      GO TO 30
50    BSTAR=CPMOD(P,B)
      IF (CPDEG(BSTAR).EQ.N) GO TO 60
      CALL CPERAS(ASTAR)
      CALL CPERAS(BSTAR)
      GO TO 30
60    CSTAR=CPRES(P,ASTAR,BSTAR)
      CALL CPERAS(ASTAR)
      CALL CPERAS(BSTAR)
70    QS=CMOD(P,Q)
      QS=CRECIP(P,QS)
      T=CPCRA(Q,C,P,CSTAR,QS,L)
      CALL PERASE(C)
      C=T
      IF (L.NE.0) CALL CPERAS(CSTAR)
80    T=PFA(P,0)
```

```
      U=IPROD(Q,T)
      CALL ERLA(T)
      CALL ERLA(Q)
      Q=U
      S=ICOMP(Q,F)
      IF (S.LT.0) GO TO 30
90    CALL ERLA(F)
      CALL ERASE(L)
      CALL ERLA(Q)
      PRES=C
      RETURN
      END

      INTEGER FUNCTION PTDIV(A,B)
      INTEGER A,ASTAR,B,BSTAR,C,CSTAR,C1,F,G,H,P,PEXP,PRIME,Q,QCAP,S,T
      INTEGER CMOD,CPCRA,CPMOD,CPTDIV,CRECIP,PCOMP
      INTEGER PFA,PNORMF,PVLIST

      COMMON /TR4/PRIME,PEXP
10    C=-1
      IF (B.EQ.0) GO TO 80
      C=0
      IF (A.EQ.0) GO TO 80
20    QCAP=PNORMF(A)
      CALL ELPOF2(QCAP,T,H)
      CALL ERLA(QCAP)
      QCAP=PNORMF(B)
      CALL ELPOF2(QCAP,T,K)
      CALL ERLA(QCAP)
      F=PEXP
      M=0
      N=0
      QCAP=PFA(1,0)
      I=PRIME
      L=INV(PVLIST(A))
30    IF (I.EQ.0) STOP
      CALL ADV(P,I)
      BSTAR=CPMOD(P,B)
      IF (BSTAR.EQ.0) GO TO 30
40    ASTAR=CPMOD(P,A)
      CSTAR=CPTDIV(P,ASTAR,BSTAR)
      IF (CSTAR.EQ.-1) GO TO 60
      CALL CPERAS(ASTAR)
      CALL CPERAS(BSTAR)
50    Q=CRECIP(P,CMOD(P,QCAP))
      C1=CPCRA(QCAP,C,P,CSTAR,Q,L)
      S=PCOMP(C,C1)
      CALL PERASE(C)
      CALL CPERAS(CSTAR)
      C=C1
      CALL IMFI(QCAP,P)
```

```
      M=M+1
      IF (S.EQ.0) GO TO 51
      N=M
      GO TO 30
51    G=M*F
      IF (G.LE.H.OR.G.LT.K+N*(F+1)) GO TO 30
      GO TO 70
60    CALL PERASE(C)
      C=-1
70    CALL ERASE(L)
      CALL ERLA(QCAP)
80    PTDIV=C
      RETURN
      END
```

## 5. A Test Program

Following is a test program which may be used to help ensure the proper implementation of this system. The program also illustrates some features common to any main program written for the application of the system. This test program results in the use of each subprogram of the system at least once.

The program begins by declaring all variables, arrays and function names to be of integer type. The UNIVAC 1108 Fortran V language provides an "implicit integer" statement which obviates the tedious manual compilation of these declarations, but they are provided here for the benefit of SAC-1 user's whose local Fortran compilers do not have this feature. Following the integer declarations are the declarations of all SAC-1 labelled common variables and a dimension statement for the array SPACE from which the available space list is constructed, and ARRAY, which is used for the generation of the list PRIME of prime numbers.

The first seven executable statements initialize those common variables requiring it, construct the available space list, and generate the list of primes. Only seven primes are generated here but many applications will require more (the regular use of about 50 is recommended); these primes are printed as the first output of the program. These statements also assign to IN and OUT the standard logical unit

numbers for card input and printed output at the local installation. The user should note that these may have to be changed.

Next the program reads in three bivariate integral polynomials, A, B and C, and prints them as output. These three polynomials, which are the only input required, are listed following the test program and preceding the listing of the output which the test program should produce.

The program next computes $A_1 = A \cdot C$, $B_1 = B \cdot C$ and $C_1 = \gcd(A_1, B_1)$, printing the results. A and B are relatively prime and C is a positive polynomial, so $C_1 = C$. Next, $A_1$ and $B_1$ are divided by C, producing $A_2 = A$ and $B_2 = B$. The resultant, D, of A and B, is computed and printed, then E, the content of C, which turns out to be 1.

Next, PDEGS is applied to C, producing $m = \partial_1(C) = 3$ and $n = \partial_2(C) = 2$. $A_1$, A and C are all reduced modulo 13, producing polynomials $A_1^*$, $A^*$ and $C^*$. $F = A_1^*/C^*$ is computed using CPQ and printed. That $F = A^*$ is verified using CPCOMP, which returns a value $S = 0$, which is printed. Finally, the monic associate of F, G, is computed and printed. Thereupon, all remaining lists are erased, the length, L, of the available space list is computed and printed, and the value of L should of course be 10000 (if each cell consists of two words).

On the UNIVAC 1108, the compilation and execution of this program, including the loading of all subprograms, requires about 10 seconds.

Test Program

```
      INTEGER AVAIL,STAK,RECORD,SYMLST,BETA,PRIME,PEXP
      INTEGER SPACE,ARRAY,IN,OUT
      INTEGER GENPR,PREAD,PMPY,PGCD,PDIV,PTDIV,PVLIST
      INTEGER PCONT,PDEGS,PRES,PFA,LENGTH
      INTEGER CPMOD,CPMON,CPQ,CPCOMP
      INTEGER A,B,C,A1,B1,A2,B2,C1,D,E,F,G
      INTEGER ASTAR,A1STAR,CSTAR,L,M,N,P,Q,S,T
      COMMON /TR1/ AVAIL,STAK,RECORD(72)
      COMMON /TR2/ SYMLST
      COMMON /TR3/ BETA
      COMMON /TR4/ PRIME,PEXP
      DIMENSION SPACE(20000),ARRAY(100)
1     CALL BEGIN(SPACE,20000)
      SYMLST=0
      BETA=2**33
      PEXP=31
      IN=5
      OUT=6
2     PRIME=GENPR(ARRAY,100,2**31+1)
      PRINT 50
      T=PRIME
3     CALL ADV(P,T)
      Q=PFA(P,0)
      CALL IWRITE(Q,OUT)
      CALL ERLA(Q)
      IF (T.NE.0) GO TO 3
4     A=PREAD(IN)
      PRINT 51
      CALL PWRITE(OUT,A)
      B=PREAD(IN)
      PRINT 52
      CALL PWRITE(OUT,B)
      C=PREAD(IN)
      PRINT 53
      CALL PWRITE(OUT,C)
5     A1=PMPY(A,C)
      PRINT 54
      CALL PWRITE(OUT,A1)
      B1=PMPY(B,C)
      PRINT 55
      CALL PWRITE(OUT,B1)
```

```
6     C1=PGCD(A1,B1)
      PRINT 56
      CALL PWRITE(OUT,C1)
      CALL PERASE(C1)
7     A2=PDIV(A1,C)
      PRINT 57
      CALL PWRITE(OUT,A2)
      CALL PERASE(A2)
      B2=PTDIV(B1,C)
      PRINT 58
      CALL PWRITE(OUT,B2)
      CALL PERASE(B2)
8     D=PRES(A,B)
      PRINT 59
      CALL PWRITE(OUT,D)
      CALL PERASE(D)
9     E=PCONT(C)
      PRINT 60
      CALL PWRITE(OUT,E)
      CALL PERASE(E)
10    L=PDEGS(C)
      CALL DECAP(N,L)
      CALL DECAP(M,L)
      PRINT 61,M,N
11    A1STAR=CPMOD(13,A1)
      ASTAR=CPMOD(13,A)
      CSTAR=CPMOD(13,C)
      F=CPQ(13,A1STAR,CSTAR)
      CALL CPERAS(A1STAR)
      CALL CPERAS(CSTAR)
      L=PVLIST(A)
      PRINT 62
      CALL CPWRIT(13,OUT,F,L)
      S=CPCOMP(ASTAR,F)
      CALL CPERAS(ASTAR)
      PRINT 63,S
12    G=CPMON(13,F)
      CALL CPERAS(F)
      PRINT 64
      CALL CPWRIT(13,OUT,G,L)
      CALL ERASE(L)
      CALL CPERAS(G)
      CALL PERASE(A1)
      CALL PERASE(B1)
      CALL PERASE(A)
      CALL PERASE(B)
      CALL PERASE(C)
      CALL ERASE(SYMLST)
      CALL ERASE(PRIME)
      L=LENGTH(AVAIL)
      PRINT 65,L
      STOP
```

```
50    FORMAT (11H PRIME LIST)
51    FORMAT (2H A)
52    FORMAT (2H B)
53    FORMAT (2H C)
54    FORMAT (3H A1)
55    FORMAT (3H B1)
56    FORMAT (3H C1)
57    FORMAT (3H A2)
58    FORMAT (3H B2)
59    FORMAT (2H D)
60    FORMAT (2H E)
61    FORMAT (3H M=,I2,6H     N=,I2)
62    FORMAT (2H F)
63    FORMAT (3H S=,I2)
64    FORMAT (2H G)
65    FORMAT (3H L=,I6)
      END
```

Test Program Input

```
((-10X**3-7X**2+3X**1+3X**0)Y**2+(-22X**3+23X**2+20X**1+12X**0)Y**1+(+26
X**3+31X**2+19X**1+2X**0)Y**0)
((+28X**3+25X**2+4X**1-18X**0)Y**2+(+7X**3+6X**2-20X**1+13X**0)Y**1+(+1X
**3+27X**2+18X**1+22X**0)Y**0)
((+11X**3+13X**2+18X**1+15X**0)Y**2+(-1X**3-30X**2*5X**1-5X**0)Y**1+(-16
X**3+10X**2+23X**1-31X**0)Y**0)
```

Test Program Output

```
PRIME LIST
+2147433659
+2147483693
+2147483713
+2147483743
+2147483777
+2147483783
+2147483813
A
((-10X**3-7X**2+3X**1+3X**0)Y**2+(-22X**3+23X**2+20X**1+12X**0)Y**1+(+26
X**3+31X**2+19X**1+2X**0)Y**0)
B
((+28X**3+25X**2+4X**1-18X**0)Y**2+(+7X**3+6X**2-20X**1+13X**0)Y**1+(+1X
**3+27X**2+18X**1+22X**0)Y**0)
C
((+11X**3+13X**2+18X**1+15X**0)Y**2+(-1X**3-30X**2-5X**1-5X**0)Y**1+(-16
X**3+10X**2+23X**1-31X**0)Y**0)
A1
((-110X**6-207X**5-238X**4-204X**3-12X**2+99X**1+45X**0)Y**4+(-232X**6+2
74X**5+380X**4+468X**3+791X**2+485X**1+165X**0)Y**3+(+468X**6+1328X**5+1
32X**4+731X**3+574X**2+137X**1-123X**0)Y**2+(+326X**6-1399X**5-1675X**4+
362X**3-443X**2-449X**1-382X**0)Y**1+(-416X**6-236X**5+604X**4+65X**3-50
4X**2-543X**1-62X**0)Y**0)
B1
((+308X**6+639X**5+873X**4+724X**3+213X**2-264X**1-270X**0)Y**4+(+49X**6
-708X**5-910X**4-271X**3+294X**2+4X**1+285X**0)Y**3+(-444X**6-26X**5+120
2X**4+1534X**3-168X**2+163X**1+823X**0)Y**2+(-113X**6-83X**5-292X**4-118
9X**3-1401X**2+719X**1-513X**0)Y**1+(-16X**6-422X**5+5X**4+418X**3-203X*
*2-52X**1-682X**0)Y**0)
C1
((+11X**3+13X**2+18X**1+15X**0)Y**2+(-1X**3-30X**2-5X**1-5X**0)Y**1+(-16
X**3+10X**2+23X**1-31X**0)Y**0)
A2
((-10X**3-7X**2+3X**1+3X**0)Y**2+(-22X**3+23X**2+20X**1+12X**0)Y**1+(+26
X**3+31X**2+19X**1+2X**0)Y**0)
B2
((+28X**3+25X**2+4X**1-18X**0)Y**2+(+7X**3+6X**2-20X**1+13X**0)Y**1+(+1X
**3+27X**2+18X**1+22X**0)Y**0)
D
(+655029X**12+3123432X**11+5232657X**10+5705944X**9+2821861X**8+518889X*
*7-901096X**6-266001X**5-5033X**4+133024X**3-111254X**2-91137X**1-50286X
**0)
E
(+1X**0)
M= 3     N= 2
F
((+3X**3+6X**2+3X**1+3X**0)Y**2+(+4X**3+10X**2+7X**1+12X**0)Y**1+(+5X**2
+6X**1+2X**0)Y**0)
S= 0
G
((+1X**3+2X**2+1X**1+1X**0)Y**2+(+10X**3+12X**2+11X**1+4X**0)Y**1+(+6X**
2+2X**1+5X**0)Y**0)
L= 10000
```

## REFERENCES

[1]. Fortran vs. Basic Fortran - A Programming Language for Information Processing on Automatic Data Processing Systems, CACM, Vol. 7, No. 10 (Oct. 1964), 591-625.

[2]. Brown, W. S., On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors, JACM, Vol. 18, No. 4 (Oct. 1971), pp. 478-504.

[3]. Brown, W. S., and J. F. Traub, On Euclid's Algorithm and the Theory of Subresultants, JACM, Vol. 18, No. 4 (Oct. 1971), pp. 505-514.

[4]. Collins, G. E., The Calculation of Multivariate Polynomial Results, JACM, Vol. 18, No. 4 (Oct. 1971), pp. 515-532.

[5]. Collins, G. E., The SAC-1 List Processing System, University of Wisconsin Computer Sciences Department Technical Report No. 129 (34 pages), July 1971.

[6]. Collins, G. E., and J. R. Pinkert, The Revised SAC-1 Integer Arithmetic System, University of Wisconsin Computing Center Technical Report No. 9 (50 pages), Nov. 1968.

[7]. Collins, G. E., The SAC-1 Polynomial System, University of Wisconsin Computer Sciences Department Technical Report No. 115 (66 pages), March 1971.

[8]. Collins, G. E., The SAC-1 Rational Function System, University of Wisconsin Computing Center Technical Report No. 8 (32 pages), September 1971.

[9]. Collins, G. E., L. E. Heindel, E. Horowitz, M. T. McClellan, and D. R. Musser, The SAC-1 Modular Arithmetic System, University of Wisconsin Technical Report No. 10 (50 pages), June 1969.

[10]. Collins, G. E., and E. Horowitz, The SAC-1 Partial Fraction Decomposition and Rational Function Integration System, University of Wisconsin Computer Sciences Department Technical Report No. 80 (47 pages), Feb. 1970.

[11]. Collins, G. E., and L. E. Heindel, The SAC-1 Polynomial Real Zero System, University of Wisconsin Computing Center Technical Report No. 18 (72 pages), Aug. 1970.

[12]. Musser, D. R., Algorithms for Polynomial Factorization, University of Wisconsin Computer Sciences Department Technical Report No. 134 (Ph.D. thesis), September 1971, 174 pages.

## INDEX OF ALGORITHMS