User-Written Maple Functions

We have saved each Maple function contained in this appendix as the text file *function_name*`.mpl`. Electronic copies of these files can be downloaded from the web site `http://www.radford.edu/npsigmon/algebrabook.html`.

Contents

Functions for Chapter 5

```
rscoeff := proc(f, x, p, a)
    local g, i, j, ng, cg, fs, field, ftable;
    fs := 2^(degree(p));
    field := Vector[row](fs);
    for i from 1 to fs-1 do
        field[i] := Powmod(a, i, p, a) mod 2:
    od:
    field[fs] := 0;
    ftable := table();
    for i from 1 to fs-2 do
        ftable[ field[i] ] := a^i:
    od:
    ftable[ field[fs-1] ] := 1;
    ftable[ field[fs] ] := 0;
    g := expand(f) mod 2;
    ng := 0;
    for j from 0 to degree(g,x) do
        cg := coeff(g, x, j):
        cg := ftable[ Rem(numer(cg), p, a) mod 2 ] /
              ftable[ Rem(denom(cg), p, a) mod 2 ];
        if degree(cg,a) < 0 then
           cg := cg * a^(fs-1);
        fi:
        if degree(cg,a) = (fs-1) then
           cg := cg/a^(fs-1);
        fi:
        ng := ng + cg*x^j:
    od:
    g := sort(ng mod 2, x);
    RETURN(g);
end:


binword := proc(cw, n, p, a, ml)
    local i, j, bvect, vs, pco, dga, binmat, binvect;
    vs :=[]:
    for i from 0 to ml do
        pco := coeff(cw, x, i):
        if pco <> 0 then
           dga := degree(pco,a):
           pco := Powmod(a, dga, p, a) mod 2:
```

```
        fi:
        for j from 0 to n-1 do
            vs := [op(vs), coeff(pco, a, j)]:
        od:
    od:
    RETURN(Vector[row](vs));
end:


bincoeff := proc(n,bmess)
    local i, j, k, bk, pcoeff, poly;
    pcoeff := []:
    bk := LinearAlgebra:-Dimension(bmess);
    i := 0;
    k := 0;
    while i < bk do
          poly := 0:
          for j from 1 to n do
              poly := poly + bmess[i+j]*a^(j-1):
          od:
          pcoeff := [op(pcoeff),poly]:
          k := k+1;
          i := k*n;
    od:
    RETURN(Vector[row]([pcoeff])):
end:


rseuclid := proc(t, f, g, z, p, a)
    local q, r, rm1, rp1, um1, u, up1, vm1, v, vp1, i;
    rm1 := sort(Expand(f) mod 2);
    r := sort(Expand(g) mod 2);
    um1 := 1;
    u := 0;
    vm1 := 0;
    v := 1;
    while degree(r,z) >= t do
          rp1 := Rem(rm1, r, z, 'q') mod 2;
          rp1 := rscoeff(rp1, z, p, a);
          q := rscoeff(q, z, p, a);
          vp1 := expand(vm1-v*q) mod 2;
          vm1 := v;
          v := sort(vp1,z);
```

```
            v := rscoeff(v, z, p, a);
            up1 := expand(um1-u*q) mod 2;
            um1 := u;
            u := sort(up1);
            u := rscoeff(u, z, p, a);
            rm1 := r;
            r := sort(rp1,z);
            print(‘Q = ‘, q, ‘  R = ‘, r, ‘  V = ‘, v,
            ‘  U = ‘, u);
    od;
    print();
    RETURN(q, r, v, u):
end:
```

Functions for Chapter 6

```
printletterfreq := proc(mess)
    local i, abet, flist, ct, j;
    flist := []:
    abet := "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    for i from 1 to length(abet) do
        ct := 0:
        for j from 1 to length(mess) do
            if evalb(substring(abet,i) = substring(mess,j))
            then
               ct := ct+1;
            fi:
        od:
        if ct > 0 then
           flist := [op(flist), [cat("Letter ", abet[i],
           " occurs ", convert(ct,string), " times"), ct]];
        fi:
    od:
    flist := sort(flist, (x,y) -> evalb(x[2] > y[2]));
    seq(lprint(flist[i][1]), i=1..nops(flist));
end:
```

Functions for Chapter 7

```
letterfreq := proc(mess)
    local i, abet, flist, ct, j;
    flist := []:
```

```
    abet := "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    for i from 1 to length(abet) do
        ct := 0:
        for j from 1 to length(mess) do
            if evalb(substring(abet,i) = substring(mess,j))
            then
               ct := ct+1;
            fi:
        od:
        flist := [op(flist),ct]:
    od:
    flist := flist/length(mess):
end:


ic := proc(c)
   local n, f, icoin;
   n := length(c);
   f := letterfreq(c);
   icoin := add(n*f[i]*(n*f[i]-1), i=1..nops(f))/(n*(n-1));
   RETURN(evalf(icoin));
end:

vigencoset := proc(kwl,mess)
    local i, monolist, pos, bstr;
    monolist := [];
    for i from 1 to kwl do
        pos := i:
        bstr := "";
        while substring(mess, pos) <> "" do
              bstr := cat(bstr, substring(mess,pos));
              pos := pos + kwl;
        od:
        monolist := [op(monolist),bstr]:
    od:
    RETURN(monolist);
end:
```

Functions for Chapter 8

```
tonumber := proc(imess)
    local sl, cn, cnl, mess, sn, ii, j, ntable;
    ntable := table(["A"=0, "B"=1, "C"=2, "D"=3, "E"=4,
```

```
    "F"=5, "G"=6, "H"=7, "I"=8, "J"=9, "K"=10, "L"=11,
    "M"=12, "N"=13, "O"=14, "P"=15, "Q"=16, "R"=17, "S"=18,
    "T"=19, "U"=20, "V"=21, "W"=22, "X"=23, "Y"=24, "Z"=25]):
    if type(imess, string) = true then
       mess := [imess];
    else
       mess := imess;
    fi:
    cnl := [];
    for j from 1 to nops(mess) do
        sl := length(mess[j]);
        cn := 0;
        for ii from 1 to sl do
            sn := ntable[substring(mess[j], ii..ii)]:
            cn := 100*cn + sn:
        od:
        cnl := [op(cnl), cn];
    od:
    if nops(cnl) = 1 then
       RETURN(op(cnl));
    else
       RETURN(cnl);
    fi:
end:


toletter := proc(num)
    local cs, cn, numl, sl, ltable, ans, anst, ii, j;
    ltable := table([0="A", 1="B", 2="C", 3="D", 4="E",
    5="F", 6="G", 7="H", 8="I", 9="J", 10="K", 11="L",
    12="M", 13="N", 14="O", 15="P", 16="Q", 17="R", 18="S",
    19="T", 20="U", 21="V", 22="W", 23="X", 24="Y", 25="Z"]);
    if type(num, integer) = true then
       numl := [num];
    else
       numl := num;
    fi:
    anst := "";
    for j from 1 to nops(numl) do
        cn := numl[j];
        sl := floor(trunc(evalf(log10(cn)))/2)+1:
        ans := "";
        for ii from 1 to sl do
```

```
            cn := cn/100;
            cs := ltable[frac(cn)*100];
            ans := cat(cs,ans);
            cn := trunc(cn);
        od:
    anst := cat(anst, ans);
    od:
    RETURN(anst);
end:
```

Functions for Chapter 9

```
epoints := proc(ec, x, ub, p)
    local ecurve, z, pct, k, i;
    pct := 0;
    for k from 0 to p-1 while pct <= ub do
        z := subs(x=k, ec) mod p;
        if z = 0 then
           pct := pct+1;
           ecurve[pct] := [k,z];
        fi:
        if z &^ ((p-1)/2) mod p = 1 then
           z := z &^ ((p+1)/4) mod p;
           ecurve[pct+1] := [k,z];
           ecurve[pct+2] := [k, -z mod p];
           pct := pct+2;
        fi:
    od:
    if pct > ub then
       pct := ub:
    fi:
    seq(ecurve[i], i=1..pct):
end:


addec := proc(le, re, c, p)
    local i, cle, cre, lambda, res, x3, y3;
    cle := le mod p;
    cre := re mod p;
    if cle = 0 or cre = 0 then
       res := cle + cre;
    elif cle[1] = cre[1] and cle[2] = -cre[2] mod p then
```

```
       res := 0;
    else
       if cle[1] = cre[1] mod p and cle[2] = cre[2] mod p
       then
          lambda := ((3*cle[1]^2+c)/2/cle[2]) mod p;
       else
          lambda := (cre[2]-cle[2])/(cre[1]-cle[1]) mod p;
       fi:
       x3 := (lambda^2-cle[1]-cre[1]) mod p;
       y3 := (lambda*(cle[1]-x3)-cle[2]) mod p;
       res := [x3,y3];
    fi:
    res;
end:


p3mod4 := proc(s)
    local t;
    t := nextprime(s);
    while t mod 4 <> 3 do
          t := nextprime(t);
    od:
    RETURN(t);
end:


elgamal := proc(alpha, e, c, p)
    local calpha, n, y;
    calpha := alpha;
    n := e;
    y := 0;
    while n > 0 do
          if irem(n, 2, 'n') = 1 then
             y := addec(calpha, y, c, p):
          fi:
          calpha := addec(calpha, calpha, c, p):
    od:
    y;
end:


tonumber := proc(imess)
    local sl, cn, cnl, mess, sn, ii, j, ntable;
    ntable := table(["A"=0, "B"=1, "C"=2, "D"=3, "E"=4,
```

```
    "F"=5, "G"=6, "H"=7, "I"=8, "J"=9, "K"=10, "L"=11,
    "M"=12, "N"=13, "O"=14, "P"=15, "Q"=16, "R"=17, "S"=18,
    "T"=19, "U"=20, "V"=21, "W"=22, "X"=23, "Y"=24, "Z"=25]):
    if type(imess, string) = true then
       mess := [imess];
    else
       mess := imess;
    fi:
    cnl := [];
    for j from 1 to nops(mess) do
        sl := length(mess[j]);
        cn := 0;
        for ii from 1 to sl do
            sn := ntable[substring(mess[j], ii..ii)]:
            cn := 100*cn + sn:
        od:
        cnl := [op(cnl), cn];
    od:
    if nops(cnl) = 1 then
       RETURN(op(cnl));
    else
       RETURN(cnl);
    fi:
end:


toletter := proc(num)
    local cs, cn, numl, sl, ltable, ans, anst, ii, j;
    ltable := table([0="A", 1="B", 2="C", 3="D", 4="E",
    5="F", 6="G", 7="H", 8="I", 9="J", 10="K", 11="L",
    12="M", 13="N", 14="O", 15="P", 16="Q", 17="R", 18="S",
    19="T", 20="U", 21="V", 22="W", 23="X", 24="Y", 25="Z"]);
    if type(num, integer) = true then
       numl := [num];
    else
       numl := num;
    fi:
    anst := "";
    for j from 1 to nops(numl) do
        cn := numl[j];
        sl := floor(trunc(evalf(log10(cn)))/2)+1:
        ans := "";
        for ii from 1 to sl do
```

```
            cn := cn/100;
            cs := ltable[frac(cn)*100];
            ans := cat(cs,ans);
            cn := trunc(cn);
        od:
    anst := cat(anst, ans);
    od:
    RETURN(anst);
end:
```

Functions for Chapter 10

```
sboxtable := proc(f,x)
    local S, u, v, d, i, j, k, rt, ct, St, iSt, pv, ffe, y,
    z, ii, jj, zv;
    v := [1, 0, 0, 0, 1, 1, 1, 1];
    S := < LinearAlgebra:-Transpose( < v > ) >;
    St := Matrix(16);
    iSt := Matrix(16);
    for i from 2 to 8 do
        v := ListTools[Rotate](v,-1);
        S := < <S>, LinearAlgebra:-Transpose( < v > ) >;
    od:
    d := degree(f,x);
    for i from 0 to 15 do
        rt := ListTools:-Reverse(convert(i, base, 2));
        while nops(rt) < d/2 do
              rt := [0,op(rt)];
        od:
        for j from 0 to 15 do
            ct := ListTools:-Reverse(convert(j, base, 2));
            while nops(ct) < d/2 do
                  ct := [0,op(ct)];
            od:
            pv := [op(rt),op(ct)];
            ffe := add(pv[degree(f)-k]*x^k,
            k=0..degree(f)-1);
            if ffe <> 0 then
               Gcdex(ffe, f, x, 'u', 'v') mod 2;
            else
               u := 0;
            fi:
```

```
            y := [];
            for k from 0 to degree(f,x)-1 do
                y := [op(y), coeff(u, x, k)];
            od:
            y := convert(y, Vector[column]);
            zv := map(m -> m mod 2, S . y + <1, 1, 0, 0, 0,
            1, 1, 0>);
            z := add(zv[k+1]*x^k, k=0..degree(f,x)-1) mod 2;
            St[i+1, j+1] := sort(z);
            ii := add(zv[d/2+k]*2^(k-1), k=1..d/2);
            jj := add(zv[k]*2^(k-1), k=1..d/2);
            iSt[ii+1, jj+1] := ffe;
        od:
    od:
    RETURN(St,iSt):
end:


sbox := proc(ffe, St, f, x)
    local i, v, r, c, d;
    v := [];
    d := degree(f,x);
    for i from 0 to d-1 do
        v := [op(v), coeff(ffe, x, i)];
    end:
    r := add(v[d/2+i]*2^(i-1), i=1..d/2);
    c := add(v[i]*2^(i-1), i=1..d/2);
    RETURN(St[r+1, c+1]);
end:


mblocks := proc(message,bl)
    local messbl, ml, mb;
    ml := length(message);
    if ml <= bl then
       messbl := cat(message, cat(seq("A", i=1..bl-ml)));
    else
       if (ml mod bl <> 0) then
          mb := iquo(ml,bl)+1;
       else
          mb := iquo(ml,bl);
       fi:
       messbl := [seq(substring(message, (i-1)*bl+1..i*bl),
```

```
        i=1..mb)];
        ml := length(messbl[mb]);
        if ml < bl then
           messbl[mb] := cat(messbl[mb], cat(seq("A",
           i=1..bl-ml)));
        fi:
    fi:
    RETURN(messbl);
end:


topoly := proc(mess,x)
    local lpoly, alphabet, sl, ltable, let, i, j;
    alphabet := "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    ltable := table():
    for i from 1 to length(alphabet) do
        ltable[ alphabet[i] ] := i-1;
    od:
    sl := length(mess);
    lpoly := [];
    for i from 1 to sl do
        let := ltable[substring(mess, i..i)];
        lpoly := [op(lpoly), sort(add(convert(let, base,
        2)[j]*x^(j-1), j=1..nops(convert(let, base, 2))))];
    od:
    RETURN(lpoly);
end:


keysched := proc(key, St, f, x)
    local r, i, j, v, w, nk, nr, m, wmat;
    nk := LinearAlgebra:-ColumnDimension(key);
    nr := nk+6;
    wmat := key;
    for i from nk+1 to 4*(nr+1) do
        v := LinearAlgebra:-Column(wmat, i-nk);
        j := i-1;
        w := LinearAlgebra:-Column(wmat,j);
        if j mod nk = 0 then
           w := convert(w,list);
           w := convert(ListTools:-Rotate(w,1),
           Vector[column]);
           w := LinearAlgebra:-Map(sbox, w, St, f, x);
```

```
            r := Rem(x^((j-nk)/nk), f, x) mod 2;
            w[1] := (w[1] + r) mod 2;
        fi:
        if nk > 6 and j mod nk = 4 then
            w := LinearAlgebra:-Map(sbox, w, St, f, x);
        fi:
        wmat := < wmat |LinearAlgebra:-Map(m -> sort(m)
        mod 2, v+w) >;
    od:
    RETURN(wmat);
end:


shiftrow := proc(M)
    local res, v, i;
    res := LinearAlgebra:-Row(M,1);
    for i from 1 to
        LinearAlgebra:-RowDimension(M)-1 do
        v := ListTools[Rotate](convert(LinearAlgebra:-Row(M,
        i+1), list), i);
        v := convert(v, Vector[row]);
        res := < res,v >;
    od:
    RETURN(res);
end:


mixcolumn := proc(M, f, x)
    local L, p, res, mp, ipoly, var;
    L := Matrix([[x,1+x,1,1], [1,x,1+x,1], [1,1,x,1+x],
    [1+x,1,1,x]]);
    mp := (p, ipoly, var) -> sort(Rem(p, ipoly, var) mod 2);
    res := LinearAlgebra:-Map(mp,
    LinearAlgebra:-Modular:-Multiply(2, L, M), f, x);
end:


invmixcolumn := proc(M, f, x)
    local iL, p, res, mp, ipoly, var;
    iL := Matrix([[x+x^2+x^3,1+x+x^3,1+x^2+x^3,1+x^3],
    [1+x^3,x+x^2+x^3,1+x+x^3,1+x^2+x^3],
    [1+x^2+x^3,1+x^3,x+x^2+x^3,1+x+x^3],
    [1+x+x^3,1+x^2+x^3,1+x^3,x+x^2+x^3]]);
    mp := (p, ipoly, var) -> sort(Rem(p, ipoly, var) mod 2);
```

```
    res := LinearAlgebra:-Map(mp,
    LinearAlgebra:-Modular:-Multiply(2, iL, M), f, x);
end:


invaddrkey := proc(M, K, f, x)
    local iL, res, mp, p, ipoly, var;
    iL := Matrix([[x+x^2+x^3,1+x+x^3,1+x^2+x^3,1+x^3],
    [1+x^3,x+x^2+x^3,1+x+x^3,1+x^2+x^3],
    [1+x^2+x^3,1+x^3,x+x^2+x^3,1+x+x^3],
    [1+x+x^3,1+x^2+x^3,1+x^3,x+x^2+x^3]]);
    mp := (p, ipoly, var) -> sort(Rem(p, ipoly, var) mod 2);
    res := LinearAlgebra:-Map(mp,
    LinearAlgebra:-Modular:-AddMultiple(2, M,
    LinearAlgebra:-Modular:-Multiply(2, iL, K)), f, x);
    RETURN(res);
end:


invshiftrow := proc(M)
    local res, v, i;
    res := LinearAlgebra:-Row(M,1);
    for i from 1 to LinearAlgebra:-RowDimension(M)-1 do
        v := ListTools[Rotate](convert(LinearAlgebra:-Row(M,
        i+1), list), -i);
        v := convert(v, Vector[row]);
        res := < res,v >;
    od:
    RETURN(res);
end:


frompoly := proc(polyv,x)
    local alphabet, cpb10, p, res;
    alphabet := "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    cpb10 := (p,x) -> if p <> 0 then
        add(coeff(p, x, i)*2^i, i=0..degree(p))
        else 0 fi;
    res := convert(LinearAlgebra:-Map(cpb10, polyv, x),
    list);
    RETURN(cat(seq(alphabet[res[i]+1], i=1..nops(res))));
end:
```

```
aesencipher := proc(message, key, St, p, x)
    local i, A, Keyin, KS, K0, Ai, Ki, B, C, Dm, nk, nr, res,
    cpb10;
    A := Matrix(4, 4, (i,j) -> topoly(message, x)[(j-1)*4 +
    i]);
    nk := length(key)/4;
    nr := nk+6;
    Keyin := Matrix(4, nk, (i,j) -> topoly(key,x)[(j-1)*4 +
    i]);
    KS := keysched(Keyin, St, p, x);
    K0 := Matrix([LinearAlgebra:-Column(KS, 1..4)]);
    Ai := addrkey(A,K0);
    for i from 1 to nr-1 do
        B := bytesub(Ai, St, p, x);
        C := shiftrow(B);
        Dm := mixcolumn(C, p, x);
        Ki := Matrix([LinearAlgebra:-Column(KS,
        4*i+1..4*(i+1))]);
        Ai := addrkey(Dm,Ki);
    od:
    B := bytesub(Ai, St, p, x);
    C := shiftrow(B);
    Ki := Matrix([LinearAlgebra:-Column(KS,
    4*nr+1..4*(nr+1))]);
    Ai := addrkey(C,Ki);
    cpb10 := (p,x) -> if p <> 0 then
        add(coeff(p, x, i)*2^i, i=0..degree(p))
        else 0 fi;
    res := convert(LinearAlgebra:-Map(cpb10, Ai, x),
    Vector[column]);
    RETURN(convert(res,list));
end:


aesdecipher := proc(ctext, key, St, iSt, p, x)
    local i, A, Ai, Keyin, K0, Ki, KS, B, C, Dm, nr, nk,
    cpb10, ml, res, alphabet;
    ml := [seq(add(convert(ctext[i],base,2)[j]*x^(j-1),
    j=1..nops(convert(ctext[i],base,2))), i=1..nops(ctext))];
    Ai := Matrix(4, 4, (i,j) -> ml[(j-1)*4 + i]);
    nk := length(key)/4;
    nr := nk+6;
    Keyin := Matrix(4, nk, (i,j) -> topoly(key,x)[(j-1)*4 +
```

```
    i]);
    KS := keysched(Keyin, St, p, x);
    Ki := Matrix([LinearAlgebra:-Column(KS,
    4*nr+1..4*(nr+1))]);
    Ai := addrkey(Ai,Ki);
    for i from nr-1 to 1 by -1 do
        Dm := bytesub(Ai, iSt, p, x);
        C := invshiftrow(Dm);
        B := invmixcolumn(C, p, x);
        Ki := Matrix([LinearAlgebra:-Column(KS,
        4*i+1..4*(i+1))]);
        Ai := invaddrkey(B, Ki, p, x);
    od:
    Ai := bytesub(Ai, iSt, p, x);
    Ai := invshiftrow(Ai);
    K0 := Matrix([LinearAlgebra:-Column(KS, 1..4)]);
    A := addrkey(Ai,K0);
    cpb10 := (p,x) -> if p <> 0 then
        add(coeff(p, x, i)*2^i, i=0..degree(p))
        else 0 fi;
    res := convert(convert(map(cpb10,
    LinearAlgebra:-Transpose(A), x), Vector[row]), list);
    alphabet := "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    RETURN(cat(seq(alphabet[res[i]+1], i=1..nops(res))));
end:
```

Functions for Chapter 11

```
cicn := proc(n, x)
    local i, d, p;
    d := sort(convert(numtheory:-divisors(n), list));
    p := 0;
    for i from 1 to nops(d) do
        p := p + numtheory:-phi(d[i])*x[d[i]]^(n/d[i]);
    od:
    p := 1/n*p;
end:


cpinv := proc(pinv, term)
   local cterm, i, st;
   cterm := pinv:
   if degree(term) <> degree(pinv) then
```

```
      cterm := 0;
   else
      st := StringTools:-Split(convert(term, string), "*");
      for i from 1 to nops(st) do
          cterm := coeff(cterm, parse(st[i]));
      od:
   fi:
   cterm;
end:


cidn := proc(n, x)
    local i, d, p;
    d := sort(convert(numtheory:-divisors(n), list));
    p := 0;
    for i from 1 to nops(d) do
        p := p + numtheory:-phi(d[i])*x[d[i]]^(n/d[i]);
    od:
    if n mod 2 = 1 then
       p := 1/(2*n)*p + 1/2*x[1]*x[2]^((n-1)/2);
    else
       p := 1/(2*n)*p + 1/4*(x[1]^2*x[2]^((n-2)/2)
           + x[2]^(n/2));
    fi:
end:
```

Functions for Chapter 12

```
cisn := proc(n,x)
   local i, k, vt, P, part, cip, partlist;
   cip := 0;
   partlist := proc(y,p)
       p[y] := p[y]+1;
   end:
   for i from 1 to combinat[numbpart](n) do
       P := Vector[row](n);
       if i = 1 then
          part := combinat[firstpart](n);
          map(partlist, part, P);
       else
          part := combinat[nextpart](part);
          map(partlist, part, P);
```

```
       fi:
       vt := 1;
       for k from 1 to n do
           if P[k] > 0 then
              vt := vt*(1/(k^P[k]*P[k]!))*x[k]^P[k];
           fi:
       od:
       cip := cip + vt;
   od:
   RETURN(cip);
end:


igraph := proc(n, x, mi)
    local P, i, k, r, s, maxv, pnum, w, part, os, es, cires,
    partlist;
    x := 'x';
    maxv := n;
    cires := 0;
    partlist := proc(y,p)
        p[y] := p[y]+1;
    end:
    os := [seq(2*i+1, i=0..trunc((n-1)/2))];
    es := [seq(2*i, i=1..trunc(n/2))];
    for pnum from 1 to combinat[numbpart](n) do
        if pnum = 1 then
           part := combinat[firstpart](n);
        else
           part := combinat[nextpart](part);
        fi:
        P := Vector[row](n);
        map(partlist, part, P);
        w := mul(1/(k^(P[k])*P[k]!), k=1..n) *
        mul((x[i/2]*x[i]^((i-2)/2))^P[i], i=es) *
        mul(x[i]^((i-1)/2*P[i]), i=os) *
        mul(x[i]^(i*binomial(P[i],2)), i=1..trunc(n/2));
        for r from 1 to n-2 do
            for s from r+1 to n-1 do
                if (P[r] > 0) and (P[s] > 0) then
                   w := w *
                   x[ilcm(r,s)]^(igcd(r,s)*P[r]*P[s]);
                   maxv := max(maxv, ilcm(r,s));
                fi:
```

```
            od:
        od:
        cires := cires + w;
    od:
    if nargs > 2 then
       mi := maxv;
    fi:
    RETURN(cires);
end:
```

Functions for Chapter 13

```
cicn := proc(n, x)
    local i, d, p;
    d := sort(convert(numtheory:-divisors(n), list));
    p := 0;
    for i from 1 to nops(d) do
        p := p + numtheory:-phi(d[i])*x[d[i]]^(n/d[i]);
    od:
    p := 1/n*p;
end:
```

```
cidn := proc(n, x)
    local i, d, p;
    d := sort(convert(numtheory:-divisors(n), list));
    p := 0;
    for i from 1 to nops(d) do
        p := p + numtheory:-phi(d[i])*x[d[i]]^(n/d[i]);
    od:
    if n mod 2 = 1 then
       p := 1/(2*n)*p + 1/2*x[1]*x[2]^((n-1)/2);
    else
       p := 1/(2*n)*p + 1/4*(x[1]^2*x[2]^((n-2)/2)
           + x[2]^(n/2));
    fi:
end:
```

```
cperm := proc(p)
    local lp, i, rl, tmp, stmp, j;
    lp := StringTools:-DeleteSpace(convert(p, string)):
    rl := [];
```

```
    tmp := [];
    i := 1;
    while lp[i] <> "" do
        if lp[i] = "(" or lp[i] = "," then
           j := i+1;
           stmp := "";
           while lp[j] <> "," and lp[j] <> ")" do
              stmp := cat(stmp, lp[j]);
              j := j + 1;
           od:
           tmp := [op(tmp), parse(stmp)]:
        fi:
        i := j;
        if lp[i] = ")" then
           rl := [op(rl), tmp]:
           tmp := []:
           i := i + 1;
        fi:
    od:
    rl;
end:


fperm := proc(p, a)
    local i, tp, j, fd, k;
    fd := 0;
    k := a;
    for i from 1 to nops(p) do
        tp := p[i];
        for j from 1 to nops(tp) - 1 do
            if k = tp[j] then
               k := tp[j+1];
               fd := 1;
               break;
            fi:
            if k = tp[nops(tp)] and fd <> 1 then
               k := tp[1];
               fd := 1;
            fi:
        od:
        if fd = 1 then
           break;
        fi:
```

```
    od:
    k;
end:


midichords := proc(fn, instr, poly, d)
   local group, i, j, k, s, res, nct, ct, eqnotes,
         notes, ntable, o, track, m;
   if add([coeffs(poly)][i], i =
      1..nops([coeffs(poly)])) = 352 then
      group := convert(GroupTheory:-Elements(
               GroupTheory:-CyclicGroup(12)), list);
   else
      group := convert(GroupTheory:-Elements(
               GroupTheory:-DihedralGroup(12)), list);
   fi:
   s := [];
   for i from 1 to nops(group) do
       s := [op(s), cperm(group[i])]
   od:
   notes := combinat:-choose(12,d);
   ntable := table():
   for i from 1 to nops(notes) do
       ntable[ notes[i] ] := i:
   od:
   nct := coeff(poly, b, d);
   ct := 0;
   eqnotes := [];
   for j from 1 to nops(notes) do
       res := [];
       for i from 1 to nops(s) do
           res := sort(ListTools:-MakeUnique([op(res),
                  ntable[sort( [seq(fperm(s[i],
                  notes[j][k]),
                  k = 1..nops(notes[j]))] )]])):
       od:
       if j = res[1] then
          ct := ct + 1:
          eqnotes := [op(eqnotes), notes[j] -
                     [seq(1, k = 1..nops(notes[j]))]];
       fi:
       if ct = nct then
          break;
```

```
       end:
    od:
    o := Array(0..11, [60, 61, 62, 63, 64, 65, 66, 67,
                       68, 69, 70, 71]);
    track := [];
    for j from 1 to d do
        track := [op(track), 77, 84, 114, 107,
                  op(ListTools:-Reverse(convert(
                  2^32 + 9*(nct) + 7, base, 256))[2..-1]),
                    0, 192, instr, seq( op([0, 144,
                    o[eqnotes[i][j]], 96, 130, 30, 144,
                    o[eqnotes[i][j]], 0]), i = 1..nct),
                    0, 255, 47, 0];
    od:
    m := [77, 84, 104, 100, 0, 0, 0, 6, 0, 1, 0, d, 0,
          128, op(track)];
    writebytes(fn, m);
    close(fn);
    eqnotes;
end:


notetoperm := proc(n)
   local c, p, s, i, k, ct, sn;
   s := Array(0..nops(n)-1);
   for i from 0 to nops(n)-1 do
       s[i] := n[i+1];
   od:
   sn := sort(n);
   ct := 0;
   p := [];
   while ct < nops(n) do
      c := [sn[1]];
      k := sn[1];
      while s[k] <> c[1] do
         c := [op(c), s[k]];
         k := s[k];
      od:
      ct := ct + nops(c);
      sn := convert(convert(sn, set)
            minus convert(c, set), list);
      if nops(c) > 1 then
         p := [op(p), c];
```

```
      fi:
   od:
   p;
end:


Tn := proc(n, p)
   local ires, ores, tmp, i, j, tn;
   ires := [];
   for i from 1 to nops(p) do
       tmp := p[i] + [seq(1, j = 1..nops(p[i]))];
       ires := [op(ires), tmp];
   od:
   tn := [];
   for i from 1 to n do
       tn := group:-mulperms(tn , [ [1, 2, 3, 4, 5, 6,
             7, 8, 9, 10, 11, 12] ]);
   od:
   ires := group:-mulperms(ires , tn);
   ores := [];
   for i from 1 to nops(ires) do
       tmp := ires[i] - [seq(1, j = 1..nops(ires[i]))];
       ores := [op(ores), tmp];
   od:
   ores;
end:


permtonote := proc(p)
   local i, j, n, tmp;
   n := Array(0..11, [0,1,2,3,4,5,6,7,8,9,10,11]);
   for i from 1 to nops(p) do
       tmp := n[p[i][1]];
       for j from 1 to nops(p[i]) - 1 do
           n[p[i][j]] := n[p[i][j+1]];
       od:
       n[p[i][nops(p[i])]] := tmp;
   od:
   convert(n, list);
end:


Ret := proc(n, p)
    local ires, ores, tmp, rn, i, j;
```

```
    ires := [];
    for i from 1 to nops(p) do
        tmp := p[i] + [seq(1, j = 1..nops(p[i]))];
        ires := [op(ires), tmp];
    od:
    rn := [];
    for i from 1 to n do
       rn := group:-mulperms(rn , [ [1,12], [2,11],
             [3,10], [4,9], [5,8], [6,7] ]);
    od:
    ires := group:-mulperms(rn, ires);
    ores := [];
    for i from 1 to nops(ires) do
        tmp := ires[i] - [seq(1,
               j = 1..nops(ires[i]))];
        ores := [op(ores), tmp];
    od:
    ores;
end:


Inv := proc(n, p0, p)
   local ires, ores, tmp, i, j, t;
   ires := [];
   for i from 1 to nops(p) do
       tmp := p[i] + [seq(1, j = 1..nops(p[i]))];
       ires := [op(ires), tmp];
   od:
   t := [];
   for i from 1 to 2*p0 + 1 do
       t := group:-mulperms([ [1, 2, 3, 4, 5, 6,
            7, 8, 9, 10, 11, 12] ], t);
   od:
   for i from 1 to n do
       ires := group:-mulperms(ires, [ [1,12], [2,11],
               [3,10], [4,9], [5,8], [6,7] ]);
       ires := group:-mulperms(ires, t);
   od:
   ores := [];
   for i from 1 to nops(ires) do
       tmp := ires[i] - [seq(1,
              j = 1..nops(ires[i]))];
       ores := [op(ores), tmp];
```

```
   od:
   ores;
end:


midinotes := proc(fn, inotes, instr, b12)
   local j, m, n, nnum, o, notes;
   n := inotes;
   nnum := nops(n);
   o := Array(0..11, [60, 61, 62, 63, 64, 65, 66, 67,
        68, 69, 70, 71]);
   if b12 = 1 then
      while nnum mod 12 <> 0 do
          n := [op(n), 0];
          nnum := nops(n);
      od:
      notes := [];
      for j from 1 to nnum/12 do
          notes := [op(notes), seq( op([0, 144, o[n[i]],
                    96, 129, 110, 144, o[n[i]], 0]),
                    i = (j-1)*12+1..j*12), 130, 30, 144,
                    o[0],0];
      od:
      m := [77, 84, 104, 100, 0, 0, 0, 6, 0, 0, 0, 1, 0,
            128, 77, 84, 114, 107,
            op(ListTools:-Reverse(convert(
            2^32 + nops(notes) + 7, base, 256))[2..-1]),
            0, 192, instr, op(notes), 0, 255, 47, 0];
   else
      m := [77, 84, 104, 100, 0, 0, 0, 6, 0, 0, 0, 1, 0,
            128, 77, 84, 114, 107,
           op(ListTools:-Reverse(convert(2^32 + 9*(nnum) + 7,
           base, 256))[2..-1]), 0, 192, instr, seq(
           op([0, 144, o[n[i]], 96, 129, 110, 144, o[n[i]],
            0]), i = 1..nnum), 0, 255, 47, 0];
   fi:
   writebytes(fn, m);
   close(fn);
end:
```