2025 Fall Southeastern Sectional Meeting
Tulane University, New Orleans, LA
October 3-5, 2025 (Friday - Sunday)
Meeting #1210

Finding identities for rank and crank of partitions and overpartitions using Maple

Rishabh Sarma
Pennsylvania State University

```
> with(qseries); with(rank); with(thetaids); with(ETA);
```

[*J2jaclist*, *Jetamake*, *Jterm2JACPROD*, *aqprod*, *briefqshelp*, *checkmult*, *checkprod*, *dilateJterm*, *dilatejaclist*, *etamake*, *etamakebeta*, *etaq*, *findcong*, *findcongbeta*, *findhom*, *findhomcombo*, *findhomcombomodp*, *findhommodp*, *findlincombo*, *findlincombomodp*, *findmaxind*, *findnonhom*, *findnonhomcombo*, *findpoly*, *findprod*, *findprod2*, *findprodcombo*, *findprodcombo2*, *jac2J*, *jac2prod*, *jac2series*, *jaclist2J*, *jaclist2JACPROD*, *jacprod*, *jacprodmake*, *lqdegree*, *lqdegree0*, *mprodmake*, *newprodmake*, *oldsift*, *prodmake*, *qbin*, *qdegree*, *qetamake*, *qfactor*, *qs2jaccombo*, *qschanges*, *qsfunctions*, *qshelp*, *qspackageversion*, *qspversion*, *quinprod*, *sift*, θ, *θ2*, *θ3*, *θ4*, *tripleprod*, *winquist*, *zqfactor*]

[*N*, *rankgen*, *rankgentop*, *ranknum*, *rankresgen*, *rankresgenb*, *rankresnum*, *rankresnumb*, *ranktablemake*]

[*Acmake*, *Bord*, *CUSPSANDWIDMAKE1*, *DivCheck*, *GETAP2getalist*, *Gamma1ModFunc*, *IndexofGamma1*, *JACCOF*, *JACP2jaclist*, *JB2jacprod*, *JL2jacprod*, *QP2*, *Scmake*, *_ETAn*, *_ETAnm*, *_etan*, *_etanm*, *checkGamma1ModFuncJL*, *cuspequiv1*, *cuspmake1*, *cuspordofJB*, *cuspordofJBoo*, *cuspordofJL*, *cuspsetinequiv1*, *cuspwid1*, *eeta*, *egeta*, *ejac*, *eta2geta*, *eta2jac*, *fpart*, *getacuspord*, *getalist2jacprod*, *getaprodcuspORDS*, *getaprodcuspord*, *gterm2jacprod*, *jac2eprod*, *jac2getacombo*, *jac2getaprod*, *jacbase*, *jacnormalid*, *jcombobase*, *lqdegree*, *lqdegree0*, *mintotORDS*, *mixedjac2jac*, *mulJAC*, *newxy*, *numcuspequiv1*, *phiset*, *printJACIDORDStable*, *processjacid*, *provemodfuncid*, *provemodfuncidBATCH*, *qjacdegree*, *rmcofjac*, *rmcofnotqjac*, *srep*, *sset*, *thetaidschanges*, *thetaidspversion*, *twistJP*, *v0*, *vinf*]

[*ETAchanges*, *ETApversion*, *Ffind*, *Fricke*, *GPmake*, *POWERPq*, *POWERPqMODP*, *POWERq*, *POWERqMODP*, *UpLB*, *cuspORD*, *cuspORDS*, *cuspORDSnotoo*, *cuspmake*, *cuspord*, *etaCOF*, *etaCONSTANT*, *etaWe*, *etacombo2ETAPRODlist*, *etacombo2ETAPRODwCONSlist*, *etamult*, *etanormalid*, *etaprodWe*, *etaprodtoqseries*, *etaprodtoqseries2*, *etaprodtoqseriesMODP*, *fanwidth*, *gamma0FORMCHECK*, *gammacheck*, *gammacheckM*, *gp2etaprod*, *jacbotstar*, *jactopstar*, *mintotGAMMA0ORDS*, *printETAIDORDStable*, *printcuspORDS*, *printcuspords*, *provemodfuncGAMMA0UpETAid*, *provemodfuncGAMMA0UpETAidBATCH*, *provemodfuncGAMMA0id*, *provemodfuncGAMMA0idBATCH*, *qetacombo*, *vetainf*, *vp*] **(1)**

The Maple QSERIES Package VERSION 1.3

[MAPLECLOUD] | [TUTORIAL] | [FUNCTIONS] | [INSTALLATION] | [DEVELOPMENT] | [LOG] | [OLD VERSIONS]

[INSTALL UF-APPS VERSION] | [UPDATE UF-APPS VERSION]

The main features are

- Conversion of q-series to infinite products of different types including eta-products and theta products.
- Factorization of a given rational function into a finite q-product if one exists.
- Generating probable algebraic relations (if they exist) among given q-series.

RELATED PACKAGES

- RANK PACKAGE
- CRANK PACKAGE
- SPT-CRANK PACKAGE
- T-CORE PACKAGE
- ETA PACKAGE
- THETAIDS PACKAGE
- RAMAROBINSIDS PACKAGE

[MAPLECLOUD] | [TUTORIAL] | [FUNCTIONS] | [INSTALLATION] | [DEVELOPMENT] | [LOG] | [OLD VERSIONS]

NOTE: The work done in developing the QSERIES package was supported in part by the NSF under grant number DMS-9870052.

QSERIES VERSION 1.3 is dated Fri Aug 12 15:07:08 EDT 2016.
Please let F. Garvan (fgarvan@ufl.edu) know of any bugs or problems.

The url of this page is **https://qseries.org/fgarvan/qmaple/qseries/index.html**.
Created by F.G. Garvan (*fgarvan@ufl.edu*) on Thursday, November 13, 1997.
Last update made Thu Feb 3 21:34:00 EST 2022.


fgarvan@ufl.edu

DEFINITIONS/TOOLS

NOTATION :

```
> aqprod(a,q,infinity)=Product(1-a*q^n,n=0..infinity);
```

$$(a,q)_\infty = \prod_{n=0}^{\infty} (1 - a\,q^n) \tag{2}$$

```
> cyc:=k->numtheory[cyclotomic](k,z);
  cycroot:=k->RootOf(cyc(k,_X)=0);
  cc:=(a,p)->cycroot(p)^modp(a,p) + cycroot(p)^(modp(-a,p));
  xfac2:=(a,d,p)->simplify(cc(d*(3*a+(p+1)/2),p) - cc(d*(3*a+(p-1)
  /2),p));
```

$$cyc := k \mapsto numtheory[cyclotomic](k,z)$$

$$cycroot := k \mapsto RootOf(cyc(k,_X) = 0)$$

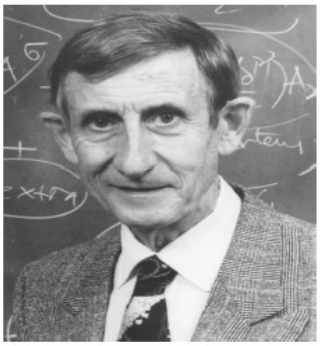
$$cc := (a,p) \mapsto cycroot(p)^{modp(a,p)} + cycroot(p)^{modp(-a,p)}$$

$$xfac2 := (a,d,p) \mapsto simplify\left(cc\left(d\cdot\left(3\cdot a + \frac{p}{2} + \frac{1}{2}\right),p\right) - cc\left(d\cdot\left(3\cdot a + \frac{p}{2} - \frac{1}{2}\right),p\right)\right) \tag{3}$$

```
> leg:=(a,p)->NumberTheory[LegendreSymbol](a,p);
  s:=p->(p^2-1)/24:
```

$$leg := (a,p) \mapsto NumberTheory[LegendreSymbol](a,p) \tag{4}$$

The rank statistic courtesy Freeman Dyson



- The Dyson rank of a partition is the largest part minus the number of parts.
- Example :

Partition	Rank
4	$4-1=3$
$3+1$	$3-2=1$
$2+2$	$2-2=0$
$2+1+1$	$2-3=-1$
$1+1+1+1$	$1-4=-3$

Mock theta functions - Dyson's prophecy

In Dyson's own words,

> *The mock theta-functions give us tantalizing hints of a grand synthesis still to be discovered. Somehow it should be possible to build them into a coherent group-theoretical structure, analogous to the structure of modular forms which Hecke built around the old theta-functions of Jacobi. This remains a challenge for the future.*
>
> Freeman Dyson
> Ramanujan Centenary Conference, 1987

- Let $N(m,n)$ denote the number of partitions of n with rank m.
- Many of Ramanujan's mock theta functions can be written in terms of $R(\zeta_p, q)$, where $R(z,q)=\sum_{n\geq 0}\sum_{m} N(m,n)z^m q^n$ is the two-variable generating function of Dyson's rank function and ζ_p is a primitive p-th root of unity.
- **Example** : $f(q)=\sum_{n\geq 0}\frac{q^{n^2}}{(-q;q)_n^2}=R(-1,q)$.

The modular structure

- **Zwegers (2002)** : Mock theta functions occur as the holomorphic part of certain real analytic modular forms.
- **Bringmann and Ono (2010)** : $R(\zeta_p, q)$ is the holomorphic part of a weak Maass form of weight $1/2$ on $\Gamma_1(576p^4)$.
- **Garvan (2019)** :
  - $\eta(p^2 z)\, R_p(\zeta_p, q)$ - the adjustment of $R(\zeta_p, q)$ is a weakly holomorphic modular form of weight 1 on $\Gamma_0(p^2) \cap \Gamma_1(p)$.
  - Considers the action of the group $\Gamma_1(p)$ on the elements of the p-dissection of $R(\zeta_p, q)$.

The dissection game of a good statistic

$$R(\zeta_p, q) = \sum_{n=0}^{\infty} \sum_{m} N(m, n)\, \zeta_p^m\, q^n = \sum_{n=0}^{\infty} \left[\sum_{k=0}^{p-1} N(k, p, n) \zeta_p^k \right] q^n = \sum_{n=0}^{\infty} \sum_{d=0}^{p-1} R_d q^{pn+d},$$

where

$$R_d = \sum_{k=0}^{p-1} N(k, p, pn + d) \zeta_p^k.$$

Then

$$\sum_{k=0}^{p-1} N(k, p, n) \zeta_p^k = N(0, p, n) - N(1, p, n) + \sum_{k=1}^{\frac{p-1}{2}} [N(k, p, n) - N(1, p, n)](\zeta_p^k + \zeta_p^{p-k})$$

and,

$$\begin{aligned} R(\zeta_p, q) &= \sum_{n=0}^{\infty} [N(0, p, n) - N(1, p, n)] q^n + \sum_{k=1}^{\frac{p-1}{2}} \sum_{n=0}^{\infty} [N(k, p, n) - N(1, p, n)] q^n (\zeta_p^k + \zeta_p^{p-k}) \\ &= \sum_{n=0}^{\infty} \sum_{d=0}^{p-1} [N(0, p, pn + d) - N(1, p, pn + d)] q^n \\ &\quad + \sum_{k=1}^{\frac{p-1}{2}} \sum_{n=0}^{\infty} \sum_{d=0}^{p-1} [N(k, p, pn + d) - N(1, p, pn + d)] q^{pn+d} (\zeta_p^k + \zeta_p^{p-k}). \end{aligned}$$

Identities for the dissection elements

One research direction with respect to these p-dissections is to find is formulas for the generating functions of $N(s,p,pn+d)-N(t,p,pn+d)$ i.e. for

$$R_{st}(d)=\sum_{n=0}^{\infty}[N(s,p,pn+d)-N(t,p,pn+d)]q^n$$

in terms of eta products/generalized eta products/modular forms and functions and/or Lambert series. The methods of finding and/or proving these formulas/identities involve two kinds of techniques.

- An adaption of the method of Atkin and Swinnerton-Dyer.
- A modular approach that involves arguments from the theory of automorphic forms.

Ramanujan's 5-dissection of $R(\zeta_5,q)$

- $N(m,n)$ = number of partitions of n with rank m.
- $R(z,q)=\sum_{n\geq 0}\sum_{m}N(m,n)z^mq^n=1+\sum_{k\geq 1}\frac{q^{k^2}}{(zq;q)_k(z^{-1}q;q)_k}$.

5-dissection rewritten in terms of generalized eta-products

$$q^{-\frac{1}{24}}\left(R(\zeta_5,q)-(\zeta_5+\zeta_5^4-2)\,\phi(q^5)+(1+2\zeta_5+2\zeta_5^4)\,q^{-2}\,\psi(q^5)\right)$$
$$=\underbrace{\frac{\eta(25z)\,\eta_{5,2}(5z)}{\eta_{5,1}(5z)^2}+\frac{\eta(25z)}{\eta_{5,1}(5z)}+(\zeta_5+\zeta_5^4)\,\frac{\eta(25z)}{\eta_{5,2}(5z)}-(\zeta_5+\zeta_5^4)\,\frac{\eta(25z)\,\eta_{5,1}(5z)}{\eta_{5,2}(5z)^2}}_{\text{weakly holomorphic modular form (with multiplier) of weight }\frac{1}{2}\text{ on the group }\Gamma_0(25)\cap\Gamma_1(5)},$$

where

$$\phi(q)=-1+\sum_{n=0}^{\infty}\frac{q^{5n^2}}{(q;q^5)_{n+1}(q^4;q^5)_n},$$

$$\psi(q)=-1+\sum_{n=0}^{\infty}\frac{q^{5n^2}}{(q^2;q^5)_{n+1}(q^3;q^5)_n}.$$

Garvan's generalization of Ramanujan's result

Theorem (Garvan, 2019)

Let $p > 3$ be prime and $1 \le a \le \frac{1}{2}(p-1)$. Define

$$R_p(\zeta_p, z) = q^{-\frac{1}{24}} R(\zeta_p, q) - \left(\frac{12}{p}\right) \sum_{a=1}^{\frac{1}{2}(p-1)} (-1)^a \left(\zeta_p^{3a+\frac{1}{2}(p+1)} + \zeta_p^{-3a-\frac{1}{2}(p+1)} + \zeta_p^{3a+\frac{1}{2}(p-1)} - \zeta_p^{-3a-\frac{1}{2}(p-1)}\right) \cdot q^{\frac{a}{2}(p-3a)-\frac{p^2}{24}} \Phi_{p,a}(q^p),$$

where

$$\Phi_{p,a}(q) = \begin{cases} \displaystyle\sum_{n=0}^{\infty} \frac{q^{pn^2}}{(q^a;q^p)_{n+1}(q^{p-a};q^p)_n}, & \text{if } 0 < 6a < p, \\ \displaystyle -1 + \sum_{n=0}^{\infty} \frac{q^{pn^2}}{(q^a;q^p)_{n+1}(q^{p-a};q^p)_n}, & \text{if } p < 6a < 3p. \end{cases}$$

Then the function $\eta(p^2 z)\, R_p(\zeta_p, z)$ is a weakly holomorphic modular form of weight 1 on the group $\Gamma_0(p^2) \cap \Gamma_1(p)$.

```
> PHIpa:=proc(p,a,T)
    local P,n:
    if 6*a<p then
       P:=add(q^(p*n^2)/aqprod(q^a,q^p,n+1)/aqprod(q^(p-a),q^p,n),
  n=0..T):
    else
       P:=-1 + add(q^(p*n^2)/aqprod(q^a,q^p,n+1)/aqprod(q^(p-a),
  q^p,n),n=0..T):
    fi:
    RETURN(P):
    end:
>
```

5-dissection of the Dyson rank of partitions

$$R(\zeta_p, q) = \sum_{n=0}^{\infty} \sum_{m} N(m,n)\, \zeta_p^m\, q^n = \sum_{n=0}^{\infty} \left[\sum_{k=0}^{p-1} N(k,p,n) \zeta_p^k \right] q^n = \sum_{n=0}^{\infty} \sum_{m=0}^{p-1} R_{p,m} q^{pn+m}$$

$$R_{5,0} = \sum_{k=0}^{4} N(k,5,5n)\zeta_5^k = A(q^5) + (\zeta_5 + \zeta_5^{-1} - 2)\, \phi(q^5),$$

$$R_{5,1} = \sum_{k=0}^{4} N(k,5,5n+1)\zeta_5^k = B(q^5),$$

$$R_{5,2} = \sum_{k=0}^{4} N(k,5,5n+2)\zeta_5^k = (\zeta_5 + \zeta_5^{-1})\, C(q^5),$$

$$R_{5,3} = \sum_{k=0}^{4} N(k,5,5n+3)\zeta_5^k = -(\zeta_5 + \zeta_5^{-1})\, D(q^5) - (\zeta_5^2 + \zeta_5^{-2} - 2)\psi(q^5)/q^5,$$

$$R_{5,4} = \sum_{k=0}^{4} N(k,5,5n+4)\zeta_5^k = 0 \quad \text{(corresponding to Dyson's rank conjecture)},$$

where

$$A(q) = \frac{(q^2, q^3, q^5; q^5)_\infty}{(q, q^4; q^5)_\infty^2},\ B(q) = \frac{(q^5; q^5)_\infty}{(q, q^4; q^5)_\infty},\ C(q) = \frac{(q^5; q^5)_\infty}{(q^2, q^3; q^5)_\infty},\ D(q) = \frac{(q, q^4, q^5; q^5)_\infty}{(q^2, q^3; q^5)_\infty^2}.$$

Modularity of elements of the dissection

Definition 4

For $p > 3$ prime, $0 \le m \le p-1$ and $1 \le d \le p-1$ define $\mathcal{K}_{p,m}(\zeta_p^d; z)$ as follows :

(i) For $m = 0$ or $(\frac{-24m}{p}) = -1$ define

$$\mathcal{K}_{p,m}(\zeta_p^d; z) := q^{m/p} \prod_{n=1}^{\infty}(1-q^{pn}) \sum_{n=\lceil \frac{1}{p}(s_p - m)\rceil}^{\infty} \left(\sum_{k=0}^{p-1} N(k,p,pn+m-s_p)\, \zeta_p^{kd} \right) q^n, \qquad (1)$$

where $s_p = \frac{1}{24}(p^2-1)$, and $q = \exp(2\pi i z)$.

(ii) For $(\frac{-24m}{p}) = 1$ define

$$\mathcal{K}_{p,m}(\zeta_p^d; z) := q^{m/p} \prod_{n=1}^{\infty}(1-q^{pn}) \left(\sum_{n=\lceil \frac{1}{p}(s_p - m)\rceil}^{\infty} \left(\sum_{k=0}^{p-1} N(k,p,pn+m-s_p)\, \zeta_p^{kd} \right) q^n \right. \qquad (2)$$

$$\left. - 4\chi_{12}(p)\,(-1)^{a+d+1} \sin(\frac{d\pi}{p}) \sin(\frac{6ad\pi}{p})\, q^{\frac{1}{p}(\frac{a}{2}(p-3a)-m)}\, \Phi_{p,a}(q) \right),$$

where $1 \le a \le \frac{1}{2}(p-1)$ has been chosen so that $-24m \equiv (6a^2) \pmod p$.

```
> Kpmd:=proc(p,m,d)
    local x,n,k,x1,a1,a,phibit,T1:
    global  finda, _fbit1, _fbit2:
    x1:=add(add(N(k,p,p*n+m-s(p))*cycroot(p)^(k*d),k=0..p-1)*q^n,
  n=ceil((s(p)-m)/p).. floor((_rtabmaxn-m+s(p))/p)):
     if m=0 or leg(-24*m,p)=-1 then
   print("CASE 1");
        x:=etaq(q,p,1000)*x1:
   else
   print("CASE 2");
        # find a
        finda:=0:
        for a1 from 1 to (p-1)/2 do
           if modp((6*a1)^2+24*m,p)=0 then
              finda:=1: a:=a1:
              #print("a=",a);
           fi:
        od:
        if finda=0 then
           ERROR("a not found in CASE 2 inside Kpmd proc");
        fi:
        T1:=floor(sqrt(_rtabmaxn/p))+3:
        phibit:=q^(1/p*(a/2*(p-3*a)-m))*PHIpa(p,a,T1):
        _fbit1:=etaq(q,p,1000)*x1:
        _fbit2:=xfac2(a,d,p)*chi12(p)*(-1)^(a)*phibit*etaq(q,p,
  1000):
        x:=_fbit1 - _fbit2:
     fi:
   RETURN(x):
   end:
```

Modularity of elements of the dissection

Theorem 5 (Garvan, 2019)

Suppose $p > 3$ prime, $0 \le m \le p-1$. Then

(i) *$\mathcal{K}_{p,0}(\zeta_p, z)$ is a weakly holomorphic modular form of weight 1 on $\Gamma_1(p)$.*

(ii) *If $1 \le m \le (p-1)$ then $\mathcal{K}_{p,m}(\zeta_p, z)$ is a weakly holomorphic modular form of weight 1 on $\Gamma(p)$. In particular,*

$$\mathcal{K}_{p,m}(\zeta_p, z)|[B]_1 = \exp\left(\frac{2\pi i b m}{p}\right) \mathcal{K}_{p,m}(\zeta_p, z),$$

for $B = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma_1(p)$.

Identity for $\mathcal{K}_{11,0}$

$$\mathcal{K}_{11,0}(\zeta_{11}, z) = (q^{11}; q^{11})_\infty \sum_{n=1}^{\infty} \left(\sum_{k=0}^{10} N(k, 11, 11n-5)\, \zeta_{11}^k \right) q^n$$

$$= \sum_{r=1}^{5} c_{11,r}\, j(11, \pi_r(\overrightarrow{n_1}), z),$$

where $\overrightarrow{n_1} = (15, -4, -2, -3, -2, -2)$, and the coefficients are :

$$c_{11,1} = -(\zeta_{11}^9 + \zeta_{11}^8 + 2\zeta_{11}^7 + \zeta_{11}^6 + \zeta_{11}^5 + 2\zeta_{11}^4 + \zeta_{11}^3 + \zeta_{11}^2 + 1),$$
$$c_{11,2} = 4\zeta_{11}^9 + \zeta_{11}^8 + 2\zeta_{11}^7 + 2\zeta_{11}^6 + 2\zeta_{11}^5 + 2\zeta_{11}^4 + \zeta_{11}^3 + 4\zeta_{11}^2 + 4,$$
$$c_{11,3} = -\zeta_{11}^9 - 2\zeta_{11}^8 + \zeta_{11}^7 - 2\zeta_{11}^6 - 2\zeta_{11}^5 + \zeta_{11}^4 - 2\zeta_{11}^3 - \zeta_{11}^2 - 3,$$
$$c_{11,4} = 2\zeta_{11}^8 + 2\zeta_{11}^7 + 2\zeta_{11}^4 + 2\zeta_{11}^3 + 3,$$
$$c_{11,5} = 2\zeta_{11}^9 + 2\zeta_{11}^8 + \zeta_{11}^7 + \zeta_{11}^4 + 2\zeta_{11}^3 + 2\zeta_{11}^2 + 1.$$

Our functions/building blocks

Dedekind eta function

$$\eta(z) := q^{\frac{1}{24}} \prod_{n=1}^{\infty}(1-q^n), \qquad q = \exp(2\pi i z),$$

A generalized eta function

$$\eta_{N,k}(z) = q^{\frac{N}{2}P_2(k/N)} \prod_{\substack{m>0 \\ m\equiv\pm k \pmod N}} (1-q^m),$$

where $z \in \mathfrak{h}$, $P_2(t) = \{t\}^2 - \{t\} + \frac{1}{6}$ is the second periodic Bernoulli polynomial, and $\{t\} = t - [t]$ is the fractional part of t.

Biagioli theta functions

$$f_{N,\rho}(z) := (-1)^{\lfloor \rho/N \rfloor} q^{(N-2\rho)^2/(8N)} (q^\rho, q^{N-\rho}, q^N; q^N)_\infty.$$

Further for a vector $\overrightarrow{n} = (n_0, n_1, n_2, \cdots, n_{\frac{p-1}{2}}) \in \mathbb{Z}^{\frac{1}{2}(p+1)}$, define

$$j(z) = j(p, \overrightarrow{n}, z) = \eta(pz)^{n_0} \prod_{k=1}^{\frac{p-1}{2}} f_{p,k}(z)^{n_k}.$$

Definition 3.4. Let $p > 3$ be prime. For $1 \le r \le \frac{1}{2}(p-1)$, we define a permutation $\pi_r : [\frac{1}{2}(p-1)] \to [\frac{1}{2}(p-1)]$, where $[\frac{1}{2}(p-1)] = \{1, 2, \cdots, \frac{1}{2}(p-1)\}$ by $\pi_r(i) = i'$ where $ri' \equiv \pm i \pmod p$.

π_r induces a permutation on $\mathbb{Z}^{\frac{1}{2}(p-1)}$. For $\overrightarrow{n} = (n_0, n_1, n_2, \cdots, n_{\frac{1}{2}(p-1)})$, $\pi_r(\overrightarrow{n})$ permutes the components to $\pi_r(\overrightarrow{n}) = (n_0, n_{\pi_r(1)}, n_{\pi_r(2)}, \cdots, n_{\pi_r(\frac{1}{2}(p-1))})$.

```
> pirp:=proc(nv,r,p)
    local n0,n1,piri,newn1,j,newnv,k:
    n0:=nv[1]:
    n1:=Array(1 .. (p-1)/2): for k from 1 to (p-1)/2 do n1[k]:=nv
  [k+1]: od:
    piri:=(r1,i)->abs(mods(i/r1,p)):
    newn1:=[seq(n1[piri(r,j)],j=1..(p-1)/2)]:
    newnv:=[n0,op(newn1)]:
  RETURN(newnv):
  end:
```

```
> mult_nv_by_eta_quot:=proc(nv,p,j)  # multiply by  (eta(p*z)/eta
  (z))^j
    local n0,newnv,k:
    n0:=nv[1]:
    newnv:=[n0 + j - j*(1 - (p-1)/2), seq(nv[k]-j,k=2..(p-1)/2+1)]:

  RETURN(newnv):
```

```
end:
```

PROVING THE IDENTITY

```
>
```

From earlier theorem, we know that K_p,m(ζp, z) is a weakly holomorphic modular form of weight 1 on Γ(p). The proof of the identity primarily involves establishing the equality using the Valence formula and showing that the RHS is also a weakly holomorphic modular form of weight 1 on Γ(p).

section, we derive and prove similar identities for $\mathcal{K}_{p,m}(\zeta_p, z)$, where $p = 11, 13, 17$ and 19 and $0 \leq m \leq p-1$.

We present a general criteria for an eta-quotient $j(p, \vec{n}, z)$ (see Definition 1.7) to be a weakly holomorphic modular form of weight 1 on $\Gamma(p)$ in the form of a theorem.

Theorem 3.1 *Let $p > 3$ be prime and suppose $\vec{n} = (n_0, n_1, n_2, \cdots, n_{\frac{1}{2}(p-1)}) \in \mathbb{Z}^{\frac{1}{2}(p+1)}$. Then $j(p, \vec{n}, z)$ is a weakly holomorphic modular form of weight 1 on $\Gamma(p)$ satisfying the modularity condition*

$$j(p, \vec{n}, z) \mid [A]_1 = \exp\left(\frac{2\pi i b m}{p}\right) j(p, \vec{n}, z)$$

for $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma_1(p)$ provided the following conditions are met:

$$(1)\ n_0 + \sum_{k=1}^{\frac{1}{2}(p-1)} n_k = 2,$$

$$(2)\ \sum_{k=1}^{\frac{1}{2}(p-1)} k^2 n_k \equiv 2m \pmod{p},$$

$$(3)\ n_0 + 3\sum_{k=1}^{\frac{1}{2}(p-1)} n_k \equiv 0 \pmod{24}.$$

```
> check_jpnz_modularity:=proc(p,m,n)
    local s1,s2,s3,k:
    #print("================================================");
    #print("n=",n);
    #print("================================================");
    s1:=add(n[k],k=1..(p+1)/2):
    s2:=n[1]+3*add(n[k],k=2..(p+1)/2):
    s3:=add( (k-1)^2*n[k],k=2..(p+1)/2):
    if s1=2 and modp(s2,24)=0 and modp(s3-2*m,p)=0 then
       RETURN(true):
    else
```

```
        print("n=",n,"s1=",s1,"s2=",s2,"=", modp(s2,24),"mod 24",
  "s3=",s3,modp(s3,p),"modp p");
        RETURN(false):
     fi:
  end:
> check_jpnz_modularity_no_detail:=proc(p,m,n)
     local s1,s2,s3,k:
     #print("================================================");
     #print("n=",n);
     #print("================================================");
     s1:=add(n[k],k=1..(p+1)/2):
     s2:=n[1]+3*add(n[k],k=2..(p+1)/2):
     s3:=add( (k-1)^2*n[k],k=2..(p+1)/2):
     if s1=2 and modp(s2,24)=0 and modp(s3-2*m,p)=0 then
        RETURN(true):
     else
        RETURN(false):
     fi:
  end:
```

We describe an algorithm for proving rank mod p identities that utilizes the Valence Formula. We apply this algorithm with the aid of the MAPLE packages THETAIDS and ETA.

Theorem 7.1 (The Valence Formula [18] (p.98)). *Let $f \neq 0$ be a modular form of weight k with respect to a subgroup Γ of finite index in $\Gamma(1) = SL_2(\mathbb{Z})$. Then*

$$ORD(f,\Gamma) = \frac{1}{12}\mu\, k, \tag{7.2}$$

where μ is index $\widehat{\Gamma}$ in $\widehat{\Gamma(1)}$,

$$ORD(f,\Gamma) := \sum_{\zeta \in R^*} ORD(f,\zeta,\Gamma),$$

R^* is a fundamental region for Γ, and

$$ORD(f;\zeta;\Gamma) = n(\Gamma;\zeta)\, ord(f;\zeta), \tag{7.3}$$

for a cusp ζ and $n(\Gamma;\zeta)$ denotes the fan width of the cusp $\zeta \pmod{\Gamma}$.

ALGORITHM STEPS

Step 1. Use Theorem 3.1 to check the three modularity conditions for each $j_{p,m,k}(z)$, $1 \le k \le r$ in the RHS of the expression (7.1). This shows that the RHS of (7.1) is a weakly holomorphic modular form of weight 1 on $\Gamma(p)$ satisfying the same modularity condition as $\mathcal{K}_{p,m}(\zeta_p, z)$ in Theorem 1.5.
Calculate orders at $i\infty$ of each generalized eta-quotient $j_{p,m,k}$.

```
> check_alg_step1:=proc(p,m,nL)
    local logset,n:
    # check that each jn-func correspoding to each nvec in nL
    # satisfies the modularity conditions of Theorem 3.1
    logset:={seq(check_jpnz_modularity(p,m,n), n in nL)}:
    if logset={true} then
       RETURN(true):
    else
       print("nL=",nL);
       for n in nL do
       print(n, check_jpnz_modularity(p,m,n));
       od:
       RETURN(false):
    fi:
  end:
```

```
> myseeds;
```

$$myseeds \tag{5}$$

```
> check_alg_step1(11,0,myseeds);
```

"n=", $myseeds$, "s1=", $myseeds_1 + myseeds_2 + myseeds_3 + myseeds_4 + myseeds_5 + myseeds_6$,
"s2=", $myseeds_1 + 3\,myseeds_2 + 3\,myseeds_3 + 3\,myseeds_4 + 3\,myseeds_5 + 3\,myseeds_6$, "=",
$myseeds_1 + 3\,myseeds_2 + 3\,myseeds_3 + 3\,myseeds_4 + 3\,myseeds_5 + 3\,myseeds_6$, "mod 24",
"s3=", $myseeds_2 + 4\,myseeds_3 + 9\,myseeds_4 + 16\,myseeds_5 + 25\,myseeds_6$, $myseeds_2$
$+ 4\,myseeds_3 + 9\,myseeds_4 + 5\,myseeds_5 + 3\,myseeds_6$, "modp p"

"nL=", $myseeds$

"n=", $myseeds$, "s1=", $myseeds_1 + myseeds_2 + myseeds_3 + myseeds_4 + myseeds_5 + myseeds_6$,
"s2=", $myseeds_1 + 3\,myseeds_2 + 3\,myseeds_3 + 3\,myseeds_4 + 3\,myseeds_5 + 3\,myseeds_6$, "=",
$myseeds_1 + 3\,myseeds_2 + 3\,myseeds_3 + 3\,myseeds_4 + 3\,myseeds_5 + 3\,myseeds_6$, "mod 24",
"s3=", $myseeds_2 + 4\,myseeds_3 + 9\,myseeds_4 + 16\,myseeds_5 + 25\,myseeds_6$, $myseeds_2$
$+ 4\,myseeds_3 + 9\,myseeds_4 + 5\,myseeds_5 + 3\,myseeds_6$, "modp p"

$myseeds, false$

$$false \tag{6}$$

Find order at infinity for each j(p,n,z)

```
> n2jlist:=proc(p,n)
    local JL,k,r:
     #Converts n-vector to j-list
     #j-list has form  [[p,k,r], ..... ] where [p,k,r] corresponds
  to JAC(k,p,infinity)^r
```

```
  JL:=[]:
  for k from 2 to (p+1)/2 do
     r:=n[k]:
     if r<>0 then
        JL:=[op(JL),[p,k-1,r]]:
     fi:
  od:
RETURN(JL): end:
#-----------------------------------------------------------
BordJterm:=(a,c,L)->Bord(L[1],L[2],a,c)*L[3]:
#-----------------------------------------------------------
BordJLIST:=(a,c,JL)->local L;add(BordJterm(a,c,L), L in JL):
#-----------------------------------------------------------
 ncuspord:=proc(p,n,a,c)
    local ep,ord1,ord2:
    #ncuspord(p,n,a,c) calculates ord(jn,a/c) where jn=j(p,n,z)
    ep:=eta(p*tau)^n[1]:
    ord1:=BordJLIST(a,c,n2jlist(p,n)):
    if c<>0 then
      ord2:=cuspord(ep,a/c):
     else
      ord2:=vetainf(ep):
    fi:
    RETURN(ord1+ord2):
 end:
 ncuspORD:=(p,n,a,c)->ncuspord(p,n,a,c)*cuspwid1(a,c,p):
#-----------------------------------------------------------
#BJL:=map(f->J2jaclist(subs(q=1,jac2J(f))), L3):
#-----------------------------------------------------------
 JLterm2n:=proc(p,JLT)
   local n,n1,k,nk,n0:
   n0:=select(L->if nops(L)=2 then true else false fi, JLT):
   if n0=[] then
     n:=[0]:
   else
     n1:=n0[1][2]:
     n:=[n1]:
   fi:
   for k from 2 to (p+1)/2 do
     nk:=select(L->if L[2]=k-1 then true else false fi, JLT):
     if nk=[] then
       n:=[op(n),0]:
     else
       n:=[op(n),nk[1][3]]:
     fi:
   od:
 RETURN(n):
 end:
#-----------------------------------------------------------
 findlowordinfnvec:=proc(p,nLIST)
   local k0,ordk0,k,ordk:
   # RETURN k=k0 so that ord(nLIST[k0],oo) = min ord(nlist[k],
oo)
   k0:=1:  ordk0:=ncuspord(p,nLIST[1],1,0):
   for k from 2 to nops(nLIST) do
     ordk:=ncuspord(p,nLIST[k],1,0):
     if ordk<=ordk0 then
```

```
          k0:=k: ordk0:=ordk:
        fi:
      od:
      RETURN(k0);
   end:
#-------------------------------------------------------------
nrat:=(n1,n2)->local k;[seq(n1[k]-n2[k],k=1..nops(n1))]:
#-------------------------------------------------------------
dividebyj0:=(nL,m)->local k;[seq(nrat(nL[k],nL[m]),k=1..nops(nL))
]:
#-------------------------------------------------------------
 n2JAC:=proc(p,n)
   local x,k,qpow:
   x:=JAC(0,p,infinity)^n[1]:
   x:=x*mul(JAC(k-1,p,infinity)^n[k],k=2..(p+1)/2):
   qpow:=ncuspord(p,n,1,0):
   RETURN(x*q^qpow):
 end:
#-------------------------------------------------------------
 checkGamma1ModFunc:=proc(p,n)
    local x,jprod,eprod,GL:
    # Check that new n-vec is a mod func on GAMMA1[p]:
    if convert(n,set)={0} then RETURN(true) else
     jprod:=n2JAC(p,n):
     eprod:=jac2eprod(jprod):
     GL:=GETAP2getalist(eprod):
     x:=Gamma1ModFunc(GL,p):
     if x=1 then RETURN(true) else RETURN(false) fi: fi: end:
```

```
> #-------------------------------------------------------------
  -------------------------------------------------------------
```

Step 2. For the cases, when $m \neq 0$, convert the eta quotients to weight 0 by dividing each by the eta-quotient having the lowest order at $i\infty$, i.e., choose $k = k_0$ such that

$$\mathrm{ORD}(j_{p,m,k_0}(z), i\infty, \Gamma_1(p)) = \min_{1\le k\le r} \mathrm{ORD}(j_{p,m,k}(z), i\infty, \Gamma_1(p)).$$

Let $j_0 = j_{p,m,k_0}(z)$. Then (7.1) has the following equivalent form:

$$\frac{\mathcal{K}_{p,m}(\zeta_p, z)}{j_0} = \sum_{k=1}^{r} c_{p,m,k} \frac{j_{p,m,k}(z)}{j_0} \tag{7.4}$$

We note that the LHS and each term on the RHS is a modular function on $\Gamma_1(p)$. When $m = 0$, we skip this step.

```
> check_alg_step2:=proc(p,m,nL)
    local n,k0,newL,logset1;
```

```
    k0:=findlowordinfnvec(p,nL):
    #print("STEP 2: k0 = ",k0);
    newL:=dividebyj0(nL,k0):
    logset1:=convert(map(nL->checkGamma1ModFunc(p,nL),newL),
             set):
    if logset1={true} then
       RETURN(newL):
    else
       ERROR("STEP 2: some new n-vec is not a modfunc on GAMMA1[p]
");
    fi:
  end:
> #-----------------------------------------------------------------
  -----------------------------------------------------------------
```

Step 3. At each cusp $s \in \mathcal{S}_p$ given in Proposition 7.2, calculate

$$\begin{aligned} &\mathrm{ORD}\left(j_{p,m,k}(z), s, \Gamma_1(p)\right) \quad \text{when} \quad m = 0, \\ &\mathrm{ORD}\left(\frac{j_{p,m,k}(z)}{j_0}, s, \Gamma_1(p)\right) \quad \text{when} \quad m \neq 0. \end{aligned}$$

for $1 \leq k \leq r$, using Proposition 7.4.

```
> check_alg_step3:=proc(p,newL)
    local CUSPSp,a,k:
    # Table of ORDS at all cusps for each func in newL
    CUSPSp:=[[1,0],[0,1],seq([1,k],k=2..(p-1)/2),seq([k,p],k=2..
  (p-1)/2)]:
    print("CUSPS:" ,CUSPSp);
    print("TABLE of ords");
    for a from 1 to nops(newL) do
    ##print(seq(ncuspord(p,newL[a],CUSPSp[k][1],CUSPSp[k][2]),k=1..
  nops(CUSPSp)));
    print(seq(ncuspORD(p,newL[a],CUSPSp[k][1],CUSPSp[k][2]),k=1..
  nops(CUSPSp)));
    od:
  RETURN():
  end:
> #-----------------------------------------------------------------
  -----------------------------------------------------------------
```

Lower bounds for the orders of K_p,m(ζp, z) at the cusps of Γ1(p)

Theorem 6.1 *Let $p \geq 3$ be a prime and $0 \leq m \leq p-1$. Then*

(i)

$$ord(\mathcal{K}_{p,m}(\zeta_p, z); 0) \begin{cases} \geq 0 & \text{if } p = 5, 7, \\ = \frac{-1}{24p}(p-5)(p-7) & \text{if } p > 7; \end{cases}$$

(ii)

$$ord(\mathcal{K}_{p,m}(\zeta_p, z); \frac{1}{n}) \begin{cases} = \frac{-3}{2p}(\frac{1}{6}(p-1)-n)(\frac{1}{6}(p+1)-n) & \text{if } 2 \leq n < \frac{1}{6}(p-1), \\ \geq 0 & \text{otherwise;} \end{cases}$$

(iii)

$$ord(\mathcal{K}_{p,m}(\zeta_p, z); \frac{n}{p}) \geq \begin{cases} \frac{1}{24p}(p^2-1) & \text{if } m = 0 \text{ or } \left(\frac{-24m}{p}\right) = -1, \\ \frac{1}{2} - \frac{3}{2p} & \text{otherwise.} \end{cases}$$

```
> Lpms:=proc(p,m,s)
       local x,n:
       if s=0 then
           if member(p,{5,7}) then x:=0 else x:=-1/24/p*(p-5)*
  (p-7): fi:
       else
           if numer(s)=1 then
              n:=denom(s):
              if n<(p-1)/6 then x:=-3/2/p*( (p-1)/6 -n)*((p+1)/2-
  n): else x:=0: fi:
            else
               n:=numer(s):
               if m=0 or numtheory[legendre](-24*m,p)=-1 then x:=
  (p^2-1)/24/p: else
                 x:=1/2-3/2/p:
               fi:
            fi:
       fi:
  RETURN(x):
  end:
  #-------------------------------------------------------------
  KjLmps:=proc(p,m,s,n)
     local x1,x2,a,c:
      #
      x1:=Lpms(p,m,s):
      a:=numer(s): c:=denom(s):
      x2:=ncuspord(p,n,a,c):
      RETURN(x1-x2):
```

```
  end:
> #-----------------------------------------------------------------
  -----------------------------------------------------------------
```

Step 4. At each cusp s, calculate the lower bound $\lambda(p, m, s)$ of

$$\begin{aligned} &\mathrm{ORD}\left(\mathcal{K}_{p,m}(\zeta_p, z), s, \Gamma_1(p)\right) \quad \text{when} \quad m = 0, \\ &\mathrm{ORD}\left(\frac{\mathcal{K}_{p,m}(\zeta_p, z)}{j_0}, s, \Gamma_1(p)\right) \quad \text{when} \quad m \neq 0. \end{aligned}$$

using Theorem 6.1. We note that value is an integer.

```
> check_alg_step4:=proc(p,m,k0,nL)
    local CUSPSpA,cuspsp,s,k:
    CUSPSpA:=[[0,1],seq([1,k],k=2..(p-1)/2),seq([k,p],k=2..(p-1)/2)
  ]:
    cuspsp:=map(L->L[1]/L[2],CUSPSpA):
    print("TABLE :");
  if m<>0 then
    print(_cusp,_LOWER_BOUND_of_ORD_of_Kpm/_j0_at_cusp);

    for s in cuspsp do
    print(_cusp=s,_LOWER_BOUND=KjLmps(p,m,s,nL[k0])*cuspwid1(numer
  (s),denom(s),p));
    od:
  else
    print(_cusp,_LOWER_BOUND_of_ORD_of_Kpm_at_cusp);

    for s in cuspsp do
    print(_cusp=s,_LOWER_BOUND=Lpms(p,m,s)*cuspwid1(numer(s),denom
  (s),p));
    od:
  fi:
  RETURN():
  end:
> #-----------------------------------------------------------------
  -----------------------------------------------------------------
```

Step 5. Calculate

$$B = \begin{cases} \displaystyle\sum_{s \in \mathcal{S}_p, s \neq i\infty} \min\left(\left\{\mathrm{ORD}\left(j_{p,m,k}(z), s, \Gamma_1(p)\right) : 1 \leq k \leq r\right\} \cup \{\lambda(p, m, s)\}\right), & \text{if } m = 0 \\ \displaystyle\sum_{s \in \mathcal{S}_p, s \neq i\infty} \min\left(\left\{\mathrm{ORD}\left(\frac{j_{p,m,k}(z)}{j_0}, s, \Gamma_1(p)\right) : 1 \leq k \leq r\right\} \cup \{\lambda(p, m, s)\}\right), & \text{if } m \neq 0 \end{cases} \tag{7.5}$$

```
>  LHSRHScuspORDtable:=proc(p,m,k0,nL,newnL)
   local b,symcusp,a,c,w,LORD,set2,RORD,LRORD,nv,n,cuspsp1,s;
      b:=0:
```

```
      cuspsp1:=cuspmake1(p):
      print("TABLE : ORD lower bounds");
      print(_cusp, _width, _ORD_LHS, _ORD_RHS,_ORD_LHS_minus_RHS);
      if m<>0 then
         n:=nL[k0]:
         for symcusp in cuspsp1 do
           if symcusp<>[1,0] then
               a:=symcusp[1]: c:=symcusp[2]:
              w:=cuspwid1(a,c,p):
               s:=a/c:
              LORD:=ceil(KjLmps(p,m,s,n)*w):
              set2:={seq(ncuspord(p,nv,a,c),nv in newnL)}:
               RORD:=ceil(min(set2)*w):
               LRORD:=ceil(min(LORD,RORD)):
              print(a/c, w, LORD,RORD, LRORD);
           fi:
         od:
    else
         for symcusp in cuspsp1 do
           if symcusp<>[1,0] then
               a:=symcusp[1]: c:=symcusp[2]:
              w:=cuspwid1(a,c,p):
               s:=a/c:
              LORD:=ceil(Lpms(p,m,s)*w):
              set2:={seq(ncuspord(p,nv,a,c),nv in newnL)}:
               RORD:=ceil(min(set2)*w):
               LRORD:=ceil(min(LORD,RORD)):
              print(a/c, w, LORD,RORD, LRORD);
           fi:
         od:
    fi:
  RETURN():
  end:
```

```
> check_alg_step5:=proc(p,m,k0,nL,newL,taboption)
    local n,b:
      if taboption then
        LHSRHScuspORDtable(p,m,k0,nL,newL);
      fi:
      n:=nL[k0]:
      b:=B_const(p,m,n,newL):
  RETURN(b):
  end:
```

```
> #-----------------------------------------------------------------
  -----------------------------------------------------------------
```

Step 6. Show

$$\mathrm{ORD}(LHS - RHS \text{ of } (7.1), i\infty, \Gamma_1(p)) \geq -B + 1 + \frac{\mu}{12} \text{ if } m = 0,$$
$$\mathrm{ORD}(LHS - RHS \text{ of } (7.4), i\infty, \Gamma_1(p)) \geq -B + 1 \text{ if } m \neq 0.$$

Here

$$\mu = \frac{1}{2}(p^2 - 1), \tag{7.6}$$

which is index of of $\widehat{\Gamma_1(p)}$ in $\widehat{\Gamma(1)}$. See [16, Thm.4.2.5, p. 106]. Then the Valence formula in Theorem 7.1 implies LHS=RHS and (7.1) is proved.

```
> minord_LHSminusRHS_at_s:=proc(p,m,s,n,nL)
      local set1,set2,set3,nv:
      if m<>0 then
        set1:={KjLmps(p,m,s,n)}:
        set2:={seq(ncuspord(p,nv,numer(s),denom(s)),nv in nL)}:
        set3:= set1 union set2:
      else
        set1:={Lpms(p,m,s)}:
        set2:={seq(ncuspord(p,nv,numer(s),denom(s)),nv in nL)}:
        set3:= set1 union set2:
      fi:
   RETURN(min(set3));
   end:
```

```
> B_const:=proc(p,m,n,nL)
      local cuspsp1,b,symcusp,a,c,minord:
      cuspsp1:=cuspmake1(p):
      b:=0:
         for symcusp in cuspsp1 do
           if symcusp<>[1,0] then
               a:=symcusp[1]: c:=symcusp[2]:
               minord:=minord_LHSminusRHS_at_s(p,m,a/c,n,nL):
               b:=b + ceil(minord*cuspwid1(a,c,p)):
            fi:
         od:
  RETURN(b):
  end:
```

```
> BARGamma1Index:=proc(N)
  local primeDIVS,p,IND1,IND0,IND01:
   #returns  the index  [barGamma(1) : barGamma[1](N)]
   primeDIVS:=select(isprime,NumberTheory[Divisors](N)):
   if N=2 then IND1:=6: else
     IND1:=N^3/2*mul(1 - 1/p^2, p in primeDIVS):
   fi:
   IND0:=N*mul(1+1/p, p in primeDIVS):
```

```
   if N=2 then IND01:=1: else
      IND01:=NumberTheory[phi](N)/2:
   fi:
   RETURN(IND0*IND01):
  end:

> check_alg_step6:=proc(p,m,k0,nL,newL)
   local symKpFL,relpm,k,relpq,topq,relpoly1,relpoly,BB;
   local GAM1IND,j0qPOW:
   global cofs,KpFL,altKp,j0q:
   cofs:=Array(1..nops(newL)):
   if m<>0 then
      j0q:=jac2series(subs(q=1,n2JAC(p,nL[k0])),300):
      j0qPOW:=m/p-ncuspord(p,nL[k0],1,0):
      altKp:=simplify(series(Kpmd(p,m,1)/j0q*q^j0qPOW,q,300)):
   else
      altKp:=simplify(series(Kpmd(p,m,1),q,300)):
   fi:
   #print(subs(cycroot(p)=zeta,simplify(series(altKp,q,5))));
   symKpFL:=map(f->n2JAC(p,f),newL);
   #print(_symKpFL=symKpFL);
   KpFL:=map(f->series(jac2series(f,300),q,301),symKpFL):
   relpm:=subs(cycroot(p)=zeta,findlincombo(altKp,KpFL,_J,q,0));
> SYMrelpm:=subs(cycroot(p)=zeta,findlincombo(altKp,KpFL,symKpFL,q,
  0));
   print("Coefficients in Kpm identity");
   for k from 1 to nops(KpFL) do
   print(_k=k, coeff(relpm,_J[k],1));
   cofs[k]:=coeff(relpm,_J[k],1);
   od:
   print("Proving and checking identity");
   relpq:=subs({zeta=cycroot(p),seq(_J[k]=KpFL[k],k=1..nops(KpFL))}
  ,relpm):
   topq:=floor(_rtabmaxn/p):
   BB:=check_alg_step5(p,m,k0,nL,newL,false):
   if topq > -BB+1 then
     relpoly1:=simplify(series(altKp-relpq,q,topq+1)):
     relpoly:=convert(relpoly1,polynom):

     if relpoly=0 then
       GAM1IND:=BARGamma1Index(p):
       print("IDENTITY CHECKED AND PROVEN");
       print("IDENTITY checked for ",_O(q^(_topq+1))=_O(q^(topq+1))
  );
       if m=0 then
          printf("and _topq + 1 > -_B + GAMMA1INDEX/12 = %a + %a =
  %a \n",-BB,GAM1IND/12,-BB+GAM1IND/12);
       else
          printf("and _topq + 1 > -_B = %a \n",-BB);
      fi:
> print(_K[p,m]=SYMrelpm);
      RETURN(cofs):
     fi:
>
   else
     print("WARNING _topq too small");
```

```
   fi:
  end:
```
Warning, (in check_alg_step6) `SYMrelpm` is implicitly declared local

```
> #-----------------------------------------------------------------
  -----------------------------------------------------------------
```

```
> do_alg_steps:=proc(p,m,nL)
    local checkstep1,k0,BB;
    global newnL:
    if m<>0 then
      printf("-------------------------------------------------
  \n");
     printf("STEP 1: Checking modularity \n");
     checkstep1:=check_alg_step1(p,m,nL):
     if checkstep1 then
        # OK
        printf("        Modularity checks\n");
        printf("-------------------------------------------------
  \n");
        printf("STEP 2: Finding k_0 and dividing by j_0\n");
          k0:=findlowordinfnvec(p,nL):
        printf("        k0 = %a \n",k0);
          newnL:=check_alg_step2(p,m,nL);
        printf("-------------------------------------------------
  \n");
        printf("STEP 3: Computing table of ORDS at all cusps for
  each function _j(p,n,z)\n");
          check_alg_step3(p,newnL);
        printf("-------------------------------------------------
  \n");
        printf("STEP 4: Computing LOWER BOUND for ORD of _Kpm/_j0
  at each cusp\n");
          check_alg_step4(p,m,k0,nL);
        printf("-------------------------------------------------
  \n");
      printf("STEP 5: Compiling LHS vs RHS ORD table at cusps and
  finding constant B\n");
          BB:=check_alg_step5(p,m,k0,nL,newnL,true);
       printf("        This implies that B = %a \n",BB);
        printf("-------------------------------------------------
  \n");
        printf("STEP 6: Proving and checking identity\n");
           check_alg_step6(p,m,k0,nL,newnL);
        printf("-------------------------------------------------
  \n");
     else
        ERROR("Step 1 FAILED");
     fi:
   else
     printf("-------------------------------------------------
  \n");
     printf("p = %a  and m = %a \n",p,m);
     printf("STEP 1: Checking modularity \n");
     checkstep1:=check_alg_step1(p,m,nL):
```

```
      if checkstep1 then
         # OK
         printf("        Modularity checks\n");
         printf("-------------------------------------------------
\n");
         printf("STEP 2: Finding k_0 and dividing by j_0\n");
         printf("        We skip this step since m = 0\n");
         printf("-------------------------------------------------
\n");
         printf("STEP 3: Computing table of ORDS at all cusps for
each function _j(p,n,z)\n");
           check_alg_step3(p,nL);
         printf("-------------------------------------------------
\n");
         printf("STEP 4: Computing LOWER BOUND for ORD of _Kpm at
each cusp\n");
           check_alg_step4(p,m,k0,nL);
         printf("-------------------------------------------------
\n");
     printf("STEP 5: Compiling LHS vs RHS ORD table at cusps and
finding constant B\n");
            BB:=check_alg_step5(p,m,k0,nL,nL,true);
            printf("        This implies that B = %a \n",BB);
         printf("-------------------------------------------------
\n");
         printf("STEP 6: Proving and checking identity\n");
            check_alg_step6(p,m,k0,nL,nL);
         printf("-------------------------------------------------
\n");
      else
         ERROR("Step 1 FAILED");
      fi:
   fi:
end:
```

FINDING AN IDENTITY

```
>
> rand5:=rand(1..4);
   findnv:=proc(L)
     local cfound,n1,n2,n3,n4,n5,snv0,mm,mm0,nv,nv0,n0,r,mm12;
     cfound:=0:
     while cfound=0 do
       n1:=-rand5(): n2:=-rand5():n3:=-rand5():n4:=-rand5():
       n5:=-rand5():
       nv0:=[n1,n2,n3,n4,n5]:
       snv0:=convert(nv0,`+`):
       n0:=2-snv0:
       nv:=[n0,op(nv0)]:
       if check_jpnz_modularity_no_detail(11,0,nv) then
         mm:=min(seq(ncuspORD(11,pirp(nv,r,11),1,0),r=1..5)):
         if mm=L then
            mm0:=min(seq(ncuspORD(11,pirp(nv,r,11),0,1),r=1..5)):
            if mm0>=-1 then
               mm12:=min(seq(ncuspORD(11,pirp(nv,r,11),1,2),r=1..5)
):
               if mm12>=-1 then
```

```
                #print(nv);
                cfound:=1:
             fi:
          fi:
       fi:
    fi:
  od:
  RETURN(nv);
end:
#-------------------------------------------------------------------
-
nvL2q:=proc(nvL,p,T)
   local nvLq:
    nvLq:=map(f->series(jac2series(n2JAC(p,f),T),q,T+1),nvL):
RETURN(nvLq):
end:
#-------------------------------------------------------------------
-
pigenr:=(seed,p)->local r: [seq(pirp(seed,r,p),r=1..(p-1)/2)]:
#-------------------------------------------------------------------
-
findseeds:=proc()
    local seed1, seed2, numfound:
    numfound:=1:
    seed1:=findnv(1):
    RETURN([seed1]):
end:
#seed1:=findnv(1):
#seed2:=findnv(2):
#seed3:=findnv(3):
#-------------------------------------------------------------------
----------------------
R11:=[seq(add((N(0,11,11*n+6)-N(k,11,11*n+6))*q^n,n=0..floor
(_rtabmaxn/11)-1),k=1..5)]:
C11:=[seq(add((M(0,11,11*n+6)-M(k,11,11*n+6))*q^n,n=0..floor
(_ctabmaxn/11)-1),k=1..5)]:
#-------------------------------------------------------------------
----------------------
RCFL:=map(f->series(f*etaq(q,11,1000)*q,q,501),R11):
#-------------------------------------------------------------------
----------------------
plantseeds:=proc(seeds,muletapows,p)
   local nvL,seed,nvs,bignvL,nvLj,L,j:
   nvL:=[]:
   for seed in seeds do
      nvs:=pigenr(seed,p):
      nvL:=[op(nvL),op(nvs)]:
   od:
   bignvL:=nvL:
   #print("numa=",nops(nvL));
   for j in muletapows do
       #print("j=",j);
       nvLj:=map(L->mult_nv_by_eta_quot(L,p,j),nvL):
       bignvL:=[op(bignvL),op(nvLj)]:
   od:
   RETURN(nvL):
end:
```

```
#-------------------------------------------------------------------
---------------------
#BIGBAS:=plantseeds([seed1,seed2,seed3],[4,8],19):
findnice:=proc(T)
  local mx,mxset:
  local fnice,c,myseeds,BIGBAS,num,BIGBASq,badrel,m,relm,k,
rmcofs,denoms;
  fnice:=0:
  c:=0:
  while fnice=0 and c<T do
myseeds:=findseeds();
print(c,"myseeds=",myseeds);
BIGBAS:=plantseeds(myseeds,[0],11):
num:=nops(BIGBAS);
BIGBASq:=nvL2q(BIGBAS,11,121):
#-------------------------------------------------------------------
---------------------
mxset:={}:
badrel:=0:
for m from 1 to 5 do
  relm:=findlincombo(RCFL[m],BIGBASq,[seq(_J[k],k=1..num)],q,0);
  if relm=NULL then
    badrel:=1:
  else
  rmcofs:={seq(coeff(relm,_J[k],1),k=1..num)}:
  denoms:=map(x->abs(denom(x)),rmcofs):
  mx:=max(op(denoms)):
  #print("m=",m,"mx=",mx,relm);
  mxset := mxset union {mx}:
  fi:
od:
  print("mxset=",mxset);
  if mxset={1} and badrel=0 then
    fnice:=1:
  fi:
c:=c+1:
od:  if fnice=1 then
print("myseeds=",myseeds);
RETURN(myseeds);  else print("nice seeds NOT found"); RETURN():
fi:
end:
```

$$rand5 := \mathbf{proc}(\,) \quad \mathbf{proc}(\,)\ \mathbf{option}\ builtin = RandNumberInterface;\ \ \mathbf{end\ proc}(6, 4, 2) + 1 \quad \mathbf{end\ proc} \tag{7}$$

```
> myseeds:=findnice(1);
```

$$0,\ "myseeds=",\ [[15, -2, -3, -4, -2, -2]]$$

$$"mxset=",\ \{1\}$$

$$"myseeds=",\ [[15, -2, -3, -4, -2, -2]]$$

$$myseeds := [[15, -2, -3, -4, -2, -2]] \tag{8}$$

```
> myseeds;
```

$$[[15, -2, -3, -4, -2, -2]] \tag{9}$$

```
> BIGBAS:=plantseeds(myseeds,[0],11):
> do_alg_steps(11,0,BIGBAS);
-------------------------------------------------
p = 11  and m = 0
STEP 1: Checking modularity
        Modularity checks
-------------------------------------------------
STEP 2: Finding k_0 and dividing by j_0
        We skip this step since m = 0
-------------------------------------------------
STEP 3: Computing table of ORDS at all cusps for each function _j(p,
n,z)
```

"CUSPS:", $[[1,0],[0,1],[1,2],[1,3],[1,4],[1,5],[2,11],[3,11],[4,11],[5,11]]$

"TABLE of ords"

$$2, -1, -1, -1, -1, -1, 3, 2, 1, 2$$
$$3, -1, -1, -1, -1, -1, 1, 2, 2, 2$$
$$2, -1, -1, -1, -1, -1, 2, 3, 2, 1$$
$$1, -1, -1, -1, -1, -1, 2, 2, 2, 3$$
$$2, -1, -1, -1, -1, -1, 2, 1, 3, 2$$

```
-------------------------------------------------
STEP 4: Computing LOWER BOUND for ORD of _Kpm at each cusp
```

"TABLE :"

$$_cusp, _LOWER_BOUND_of_ORD_of_Kpm_at_cusp$$
$$_cusp = 0, _LOWER_BOUND = -1$$
$$_cusp = \frac{1}{2}, _LOWER_BOUND = 0$$
$$_cusp = \frac{1}{3}, _LOWER_BOUND = 0$$
$$_cusp = \frac{1}{4}, _LOWER_BOUND = 0$$
$$_cusp = \frac{1}{5}, _LOWER_BOUND = 0$$
$$_cusp = \frac{2}{11}, _LOWER_BOUND = \frac{5}{11}$$
$$_cusp = \frac{3}{11}, _LOWER_BOUND = \frac{5}{11}$$
$$_cusp = \frac{4}{11}, _LOWER_BOUND = \frac{5}{11}$$
$$_cusp = \frac{5}{11}, _LOWER_BOUND = \frac{5}{11}$$

```
-------------------------------------------------
STEP 5: Compiling LHS vs RHS ORD table at cusps and finding constant
B
```

"TABLE : ORD lower bounds"

$$_cusp,\ _width,\ _ORD_LHS,\ _ORD_RHS,\ _ORD_LHS_minus_RHS$$

$$0,\ 11,\ -1,\ -1,\ -1$$

$$\frac{1}{2},\ 11,\ 0,\ -1,\ -1$$

$$\frac{1}{3},\ 11,\ 0,\ -1,\ -1$$

$$\frac{1}{4},\ 11,\ 0,\ -1,\ -1$$

$$\frac{1}{5},\ 11,\ 0,\ -1,\ -1$$

$$\frac{2}{11},\ 1,\ 1,\ 1,\ 1$$

$$\frac{3}{11},\ 1,\ 1,\ 1,\ 1$$

$$\frac{4}{11},\ 1,\ 1,\ 1,\ 1$$

$$\frac{5}{11},\ 1,\ 1,\ 1,\ 1$$

This implies that B = -1

STEP 6: Proving and checking identity

"CASE 1"

"Coefficients in Kpm identity"

$$_k=1,\ -\zeta^9-2\,\zeta^8+\zeta^7-2\,\zeta^6-2\,\zeta^5+\zeta^4-2\,\zeta^3-\zeta^2-3$$

$$_k=2,\ 2\,\zeta^9+2\,\zeta^8+\zeta^7+\zeta^4+2\,\zeta^3+2\,\zeta^2+1$$

$$_k=3,\ 4\,\zeta^9+\zeta^8+2\,\zeta^7+2\,\zeta^6+2\,\zeta^5+2\,\zeta^4+\zeta^3+4\,\zeta^2+4$$

$$_k=4,\ -\zeta^9-\zeta^8-2\,\zeta^7-\zeta^6-\zeta^5-2\,\zeta^4-\zeta^3-\zeta^2-1$$

$$_k=5,\ 2\,\zeta^8+2\,\zeta^7+2\,\zeta^4+2\,\zeta^3+3$$

"Proving and checking identity"

"IDENTITY CHECKED AND PROVEN"

"IDENTITY checked for ", $_O\left(q^{-topq+1}\right)=_O\left(q^{273}\right)$

and _topq + 1 > -_B + GAMMA1INDEX/12 = 1 + 5 = 6

$$_K_{11,\,0}=-\frac{\left(\zeta^9+\zeta^8+2\,\zeta^7+\zeta^6+\zeta^5+2\,\zeta^4+\zeta^3+\zeta^2+1\right)JAC(0,\,11,\,\infty)^{15}\,q}{JAC(1,\,11,\,\infty)^4\,JAC(2,\,11,\,\infty)^2\,JAC(3,\,11,\,\infty)^3\,JAC(4,\,11,\,\infty)^2\,JAC(5,\,11,\,\infty)^2}$$

$$-\frac{\left(-4\,\zeta^9-\zeta^8-2\,\zeta^7-2\,\zeta^6-2\,\zeta^5-2\,\zeta^4-\zeta^3-4\,\zeta^2-4\right)JAC(0,\,11,\,\infty)^{15}\,q^2}{JAC(1,\,11,\,\infty)^2\,JAC(2,\,11,\,\infty)^4\,JAC(3,\,11,\,\infty)^2\,JAC(4,\,11,\,\infty)^2\,JAC(5,\,11,\,\infty)^3}$$

$$
\begin{vmatrix}
-\dfrac{\left(\zeta^9+2\,\zeta^8-\zeta^7+2\,\zeta^6+2\,\zeta^5-\zeta^4+2\,\zeta^3+\zeta^2+3\right)JAC(0,11,\infty)^{15}\,q^2}{JAC(1,11,\infty)^2\,JAC(2,11,\infty)^3\,JAC(3,11,\infty)^4\,JAC(4,11,\infty)^2\,JAC(5,11,\infty)^2} \\
-\dfrac{\left(-2\,\zeta^8-2\,\zeta^7-2\,\zeta^4-2\,\zeta^3-3\right)JAC(0,11,\infty)^{15}\,q^2}{JAC(1,11,\infty)^3\,JAC(2,11,\infty)^2\,JAC(3,11,\infty)^2\,JAC(4,11,\infty)^4\,JAC(5,11,\infty)^2} \\
-\dfrac{\left(-2\,\zeta^9-2\,\zeta^8-\zeta^7-\zeta^4-2\,\zeta^3-2\,\zeta^2-1\right)JAC(0,11,\infty)^{15}\,q^3}{JAC(1,11,\infty)^2\,JAC(2,11,\infty)^2\,JAC(3,11,\infty)^2\,JAC(4,11,\infty)^3\,JAC(5,11,\infty)^4} \\
\text{----------}
\end{vmatrix}
$$