Q=3

#H3

kk=-17

for q in prime\_range(3,4):

if abs(kk)!=q:

L=[]

for i in range(len(divisors(396))):

if Integers(2)(divisors(396)[i])==0 and is\_prime(divisors(396)[i]+1) and divisors(396)[i]+1!=abs(kk) and divisors(396)[i]+1!=q and divisors(396)[i]+1<200:

L.append(divisors(396)[i]+1)

print(q,'L=',L)

#求cl

f = Newforms(2^5\*abs(kk), names='a')

for i in range(6):

C=f[i].padded\_list(200)#求系数

clf=[]

for j in range(len(L)):

l=L[j]

clf.append(C[l])

print('clf=',clf)

l=5

cl=clf[0]

BS=[1,2,3,4]

BT=[2, 1, 2, 0]

Est=[]

F11=Integers(11)

for i in range(len(BS)):

s=BS[i]

t=BT[i]

E=[0,2\*t,0,s^11,0]#代入s和t得到的椭圆曲线

Est.append(E)

#print('Est',Est)#根据s和t的值得到的j=1时所有的椭圆曲线

C1S=[]

C1T=[]

for j in range(len(Est)):

F=GF(l)

Est1=EllipticCurve(F,Est[j])

alE=l+1-Est1.cardinality()#计算al（Est1）

result=F11(alE-cl)

if result==0:#判断Norm条件

C1S.append(BS[j])

C1T.append(BT[j])

#print(Est[m])#得到满足Norm条件的曲线以及对应的s和t的值

#print("C1S",C1S,"C1T",C1T)

D1=[]

for i in range(len(C1S)):

s=C1S[i]

t=C1T[i]

d=Integers(l)(t^2-s^11)

D1.append(d)

D1=sorted(set(D1),key=D1.index)

#print("D1",D1)

G=[]

for b in range(396):

if Integers(4)(kk\*q^b)==1 and Integers(11)(b)==0 and Integers(l)(kk\*q^b) in D1:

G.append(b)

print("G=",G)

3 L= [5, 7, 13, 19, 23, 37, 67, 199]

clf= [2, -2, 2, 4, -2, 10, -16, -6]

G= []

clf= [4, 4, 2, -8, 8, -8, -4, -24]

G= []

clf= [0, -2, 2, -4, -6, 4, -4, -18]

G= []

clf= [0, 2, 2, 4, 6, 4, 4, 18]

G= []

clf= [2, 2, 2, -4, 2, 10, 16, 6]

G= []

clf= [4, -4, 2, 8, -8, -8, 4, 24]

G= []

Q=5

#H3

kk=-17

for q in prime\_range(5,7):

if abs(kk)!=q:

L=[]

for i in range(len(divisors(396))):

if Integers(2)(divisors(396)[i])==0 and is\_prime(divisors(396)[i]+1) and divisors(396)[i]+1!=abs(kk) and divisors(396)[i]+1!=q and divisors(396)[i]+1<200:

L.append(divisors(396)[i]+1)

print(q,'L=',L)

#求cl

f = Newforms(2^5\*abs(kk), names='a')

for i in range(6):

C=f[i].padded\_list(200)#求系数

clf=[]

for j in range(len(L)):

l=L[j]

clf.append(C[l])

print('clf=',clf)

l=3

cl=clf[0]

BS=[2]

BT=[0]

Est=[]

F11=Integers(11)

for i in range(len(BS)):

s=BS[i]

t=BT[i]

E=[0,2\*t,0,s^11,0]#代入s和t得到的椭圆曲线

Est.append(E)

#print('Est',Est)#根据s和t的值得到的j=1时所有的椭圆曲线

C1S=[]

C1T=[]

for j in range(len(Est)):

F=GF(l)

Est1=EllipticCurve(F,Est[j])

alE=l+1-Est1.cardinality()#计算al（Est1）

result=F11(alE-cl)

if result==0:#判断Norm条件

C1S.append(BS[j])

C1T.append(BT[j])

#print(Est[m])#得到满足Norm条件的曲线以及对应的s和t的值

#print("C1S",C1S,"C1T",C1T)

D1=[]

for i in range(len(C1S)):

s=C1S[i]

t=C1T[i]

d=Integers(l)(t^2-s^11)

D1.append(d)

D1=sorted(set(D1),key=D1.index)

#print("D1",D1)

G=[]

for b in range(396):

if Integers(4)(kk\*q^b)==1 and Integers(11)(b)==0 and Integers(l)(kk\*q^b) in D1:

G.append(b)

print("G=",G)

5 L= [3, 7, 13, 19, 23, 37, 67, 199]

clf= [-2, -2, 2, 4, -2, 10, -16, -6]

G= []

clf= [-2, 4, 2, -8, 8, -8, -4, -24]

G= []

clf= [0, -2, 2, -4, -6, 4, -4, -18]

G= []

clf= [0, 2, 2, 4, 6, 4, 4, 18]

G= []

clf= [2, 2, 2, -4, 2, 10, 16, 6]

G= []

clf= [2, -4, 2, 8, -8, -8, 4, 24]

G= []

Q=7

#H3

kk=-17

for q in prime\_range(7,8):

if abs(kk)!=q:

L=[]

for i in range(len(divisors(396))):

if Integers(2)(divisors(396)[i])==0 and is\_prime(divisors(396)[i]+1) and divisors(396)[i]+1!=abs(kk) and divisors(396)[i]+1!=q and divisors(396)[i]+1<200:

L.append(divisors(396)[i]+1)

print(q,'L=',L)

#求cl

f = Newforms(2^5\*abs(kk), names='a')

for i in range(6):

C=f[i].padded\_list(200)#求系数

clf=[]

for j in range(len(L)):

l=L[j]

clf.append(C[l])

print('clf=',clf)

l=3

cl=clf[0]

BS=[2]

BT=[0]

Est=[]

F11=Integers(11)

for i in range(len(BS)):

s=BS[i]

t=BT[i]

E=[0,2\*t,0,s^11,0]#代入s和t得到的椭圆曲线

Est.append(E)

#print('Est',Est)#根据s和t的值得到的j=1时所有的椭圆曲线

C1S=[]

C1T=[]

for j in range(len(Est)):

F=GF(l)

Est1=EllipticCurve(F,Est[j])

alE=l+1-Est1.cardinality()#计算al（Est1）

result=F11(alE-cl)

if result==0:#判断Norm条件

C1S.append(BS[j])

C1T.append(BT[j])

#print(Est[m])#得到满足Norm条件的曲线以及对应的s和t的值

#print("C1S",C1S,"C1T",C1T)

D1=[]

for i in range(len(C1S)):

s=C1S[i]

t=C1T[i]

d=Integers(l)(t^2-s^11)

D1.append(d)

D1=sorted(set(D1),key=D1.index)

#print("D1",D1)

G=[]

for b in range(396):

if Integers(4)(kk\*q^b)==1 and Integers(11)(b)==0 and Integers(l)(kk\*q^b) in D1:

G.append(b)

print("G=",G)

5 L= [3, 7, 13, 19, 23, 37, 67, 199]

clf= [-2, -2, 2, 4, -2, 10, -16, -6]

G= []

clf= [-2, 4, 2, -8, 8, -8, -4, -24]

G= []

clf= [0, -2, 2, -4, -6, 4, -4, -18]

G= []

clf= [0, 2, 2, 4, 6, 4, 4, 18]

G= []

clf= [2, 2, 2, -4, 2, 10, 16, 6]

G= []

clf= [2, -4, 2, 8, -8, -8, 4, 24]

G= []

11≤q≤31

#H3

kk=-17

for q in prime\_range(8,32):

if abs(kk)!=q:

L=[]

for i in range(len(divisors(396))):

if Integers(2)(divisors(396)[i])==0 and is\_prime(divisors(396)[i]+1) and divisors(396)[i]+1!=abs(kk) and divisors(396)[i]+1!=q and divisors(396)[i]+1<200:

L.append(divisors(396)[i]+1)

print(q,'L=',L)

#求cl

f = Newforms(2^5\*abs(kk), names='a')

for i in range(6):

C=f[i].padded\_list(200)#求系数

clf=[]

for j in range(len(L)):

l=L[j]

clf.append(C[l])

print('clf=',clf)

l=3

cl=clf[0]

BS=[2]

BT=[0]

Est=[]

F11=Integers(11)

for i in range(len(BS)):

s=BS[i]

t=BT[i]

E=[0,2\*t,0,s^11,0]#代入s和t得到的椭圆曲线

Est.append(E)

#print('Est',Est)#根据s和t的值得到的j=1时所有的椭圆曲线

C1S=[]

C1T=[]

for j in range(len(Est)):

F=GF(l)

Est1=EllipticCurve(F,Est[j])

alE=l+1-Est1.cardinality()#计算al（Est1）

result=F11(alE-cl)

if result==0:#判断Norm条件

C1S.append(BS[j])

C1T.append(BT[j])

#print(Est[m])#得到满足Norm条件的曲线以及对应的s和t的值

#print("C1S",C1S,"C1T",C1T)

D1=[]

for i in range(len(C1S)):

s=C1S[i]

t=C1T[i]

d=Integers(l)(t^2-s^11)

D1.append(d)

D1=sorted(set(D1),key=D1.index)

#print("D1",D1)

G3=[]

for b in range(396):

if Integers(4)(kk\*q^b)==1 and Integers(11)(b)==0 and Integers(l)(kk\*q^b) in D1:

G3.append(b)

print("G3=",G3)

l=5

cl=clf[1]

BS=[1, 2, 3, 4]

BT=[2, 1, 2, 0]

Est=[]

F11=Integers(11)

for i in range(len(BS)):

s=BS[i]

t=BT[i]

E=[0,2\*t,0,s^11,0]#代入s和t得到的椭圆曲线

Est.append(E)

#print('Est',Est)#根据s和t的值得到的j=1时所有的椭圆曲线

C1S=[]

C1T=[]

for j in range(len(Est)):

F=GF(l)

Est1=EllipticCurve(F,Est[j])

alE=l+1-Est1.cardinality()#计算al（Est1）

result=F11(alE-cl)

if result==0:#判断Norm条件

C1S.append(BS[j])

C1T.append(BT[j])

#print(Est[j])#得到满足Norm条件的曲线以及对应的s和t的值

#print("C1S",C1S,"C1T",C1T)

D1=[]

for i in range(len(C1S)):

s=C1S[i]

t=C1T[i]

d=Integers(l)(t^2-s^11)

D1.append(d)

#print("D1",D1)

G5=[]

for k in range(len(G3)):

b=G3[k]

if Integers(4)(kk\*q^b)==1 and Integers(11)(b)==0 and Integers(l)(kk\*q^b) in D1:

G5.append(b)

print("G5",G5)

l=7

cl=clf[2]

BS=[1, 2, 3, 4,5,6]

BT=[2, 4, 0, 1, 0, 0]

Est=[]

F11=Integers(11)

for i in range(len(BS)):

s=BS[i]

t=BT[i]

E=[0,2\*t,0,s^11,0]#代入s和t得到的椭圆曲线

Est.append(E)

#print('Est',Est)#根据s和t的值得到的j=1时所有的椭圆曲线

C1S=[]

C1T=[]

for j in range(len(Est)):

F=GF(l)

Est1=EllipticCurve(F,Est[j])

alE=l+1-Est1.cardinality()#计算al（Est1）

result=F11(alE-cl)

if result==0:#判断Norm条件

C1S.append(BS[j])

C1T.append(BT[j])

#print(Est[j])#得到满足Norm条件的曲线以及对应的s和t的值

#print("C1S",C1S,"C1T",C1T)

D1=[]

for i in range(len(C1S)):

s=C1S[i]

t=C1T[i]

d=Integers(l)(t^2-s^11)

D1.append(d)

#print("D1",D1)

G7=[]

for k in range(len(G5)):

b=G5[k]

if Integers(4)(kk\*q^b)==1 and Integers(11)(b)==0 and Integers(l)(kk\*q^b) in D1:

G7.append(b)

print("G7",G7)

11 L= [3, 5, 7, 13, 19, 23, 37, 67, 199]

clf= [-2, 2, -2, 2, 4, -2, 10, -16, -6]

G3= []

G5 []

G7 []

clf= [-2, 4, 4, 2, -8, 8, -8, -4, -24]

G3= []

G5 []

G7 []

clf= [0, 0, -2, 2, -4, -6, 4, -4, -18]

G3= []

G5 []

G7 []

clf= [0, 0, 2, 2, 4, 6, 4, 4, 18]

G3= []

G5 []

G7 []

clf= [2, 2, 2, 2, -4, 2, 10, 16, 6]

G3= []

G5 []

G7 []

clf= [2, 4, -4, 2, 8, -8, -8, 4, 24]

G3= []

G5 []

G7 []

13 L= [3, 5, 7, 19, 23, 37, 67, 199]

clf= [-2, 2, -2, 4, -2, 10, -16, -6]

G3= []

G5 []

G7 []

clf= [-2, 4, 4, -8, 8, -8, -4, -24]

G3= []

G5 []

G7 []

clf= [0, 0, -2, -4, -6, 4, -4, -18]

G3= []

G5 []

G7 []

clf= [0, 0, 2, 4, 6, 4, 4, 18]

G3= []

G5 []

G7 []

clf= [2, 2, 2, -4, 2, 10, 16, 6]

G3= []

G5 []

G7 []

clf= [2, 4, -4, 8, -8, -8, 4, 24]

G3= []

G5 []

G7 []

19 L= [3, 5, 7, 13, 23, 37, 67, 199]

clf= [-2, 2, -2, 2, -2, 10, -16, -6]

G3= []

G5 []

G7 []

clf= [-2, 4, 4, 2, 8, -8, -4, -24]

G3= []

G5 []

G7 []

clf= [0, 0, -2, 2, -6, 4, -4, -18]

G3= [11, 33, 55, 77, 99, 121, 143, 165, 187, 209, 231, 253, 275, 297, 319, 341, 363, 385]

G5 [11, 33, 55, 77, 99, 121, 143, 165, 187, 209, 231, 253, 275, 297, 319, 341, 363, 385]

G7 []

clf= [0, 0, 2, 2, 6, 4, 4, 18]

G3= [11, 33, 55, 77, 99, 121, 143, 165, 187, 209, 231, 253, 275, 297, 319, 341, 363, 385]

G5 [11, 33, 55, 77, 99, 121, 143, 165, 187, 209, 231, 253, 275, 297, 319, 341, 363, 385]

G7 []

clf= [2, 2, 2, 2, 2, 10, 16, 6]

G3= []

G5 []

G7 []

clf= [2, 4, -4, 2, -8, -8, 4, 24]

G3= []

G5 []

G7 []

23 L= [3, 5, 7, 13, 19, 37, 67, 199]

clf= [-2, 2, -2, 2, 4, 10, -16, -6]

G3= []

G5 []

G7 []

clf= [-2, 4, 4, 2, -8, -8, -4, -24]

G3= []

G5 []

G7 []

clf= [0, 0, -2, 2, -4, 4, -4, -18]

G3= []

G5 []

G7 []

clf= [0, 0, 2, 2, 4, 4, 4, 18]

G3= []

G5 []

G7 []

clf= [2, 2, 2, 2, -4, 10, 16, 6]

G3= []

G5 []

G7 []

clf= [2, 4, -4, 2, 8, -8, 4, 24]

G3= []

G5 []

G7 []

29 L= [3, 5, 7, 13, 19, 23, 37, 67, 199]

clf= [-2, 2, -2, 2, 4, -2, 10, -16, -6]

G3= []

G5 []

G7 []

clf= [-2, 4, 4, 2, -8, 8, -8, -4, -24]

G3= []

G5 []

G7 []

clf= [0, 0, -2, 2, -4, -6, 4, -4, -18]

G3= []

G5 []

G7 []

clf= [0, 0, 2, 2, 4, 6, 4, 4, 18]

G3= []

G5 []

G7 []

clf= [2, 2, 2, 2, -4, 2, 10, 16, 6]

G3= []

G5 []

G7 []

clf= [2, 4, -4, 2, 8, -8, -8, 4, 24]

G3= []

G5 []

G7 []

31 L= [3, 5, 7, 13, 19, 23, 37, 67, 199]

clf= [-2, 2, -2, 2, 4, -2, 10, -16, -6]

G3= []

G5 []

G7 []

clf= [-2, 4, 4, 2, -8, 8, -8, -4, -24]

G3= []

G5 []

G7 []

clf= [0, 0, -2, 2, -4, -6, 4, -4, -18]

G3= [11, 33, 55, 77, 99, 121, 143, 165, 187, 209, 231, 253, 275, 297, 319, 341, 363, 385]

G5 [11, 33, 55, 77, 99, 121, 143, 165, 187, 209, 231, 253, 275, 297, 319, 341, 363, 385]

G7 []

clf= [0, 0, 2, 2, 4, 6, 4, 4, 18]

G3= [11, 33, 55, 77, 99, 121, 143, 165, 187, 209, 231, 253, 275, 297, 319, 341, 363, 385]

G5 [11, 33, 55, 77, 99, 121, 143, 165, 187, 209, 231, 253, 275, 297, 319, 341, 363, 385]

G7 []

clf= [2, 2, 2, 2, -4, 2, 10, 16, 6]

G3= []

G5 []

G7 []

clf= [2, 4, -4, 2, 8, -8, -8, 4, 24]

G3= []

G5 []

G7 []