Maassive Computation

```
The following polynomial $f$ has splitting field a totally real
$A_4$ extension of the rationals.  It is defined below for use in
GP/PARI.
```

The following polynomial f has splitting field a totally real A_4 extension of the rationa

```
f=x^4-x^3-7*x^2+2*x+9;; f
```

```
x^4 - x^3 - 7*x^2 + 2*x + 9
```

```
We initialize a number field $K$, as a field generated by a root
of $f$.
```

We initialize a number field K, as a field generated by a root of f.

```
K=bnfinit(f);;
```

```
We compute the narrow class group of $K$.  It has order $2$,
fortunately!
```

We compute the narrow class group of K. It has order 2, fortunately!

```
bnfnarrow(K)
```

```
[2, [2], [[38, 0, 0, 29; 0, 38, 0, 33; 0, 0, 38, 16; 0, 0, 0, 1
```

```
We compute the roots of $f$, approximately, using PARI.
```

We compute the roots of f, approximately, using PARI.

```
polroots(f)
```

```
[-1.9178332447254460402367998426967616253 + 0.E-38*I,
-1.2438940346759207625691712156145068077 + 0.E-38*I,
1.3343195010635980117912682546396712746 + 0.E-38*I,
2.8274077783377687910147028036715971583 + 0.E-38*I]~
```

```
Now we switch to using SAGE, defining $M_1$, $M_2$, $M_3$, $M_4$,
as four subfields of the real numbers, obtained from the four
distinct embeddings of $K$ into $R$.
```

Now we switch to using SAGE, defining M_1, M_2, M_3, M_4, as four subfields of the rea

```
M1.<a1> = NumberField(x^4 - x^3 - 7*x^2 + 2*x + 9, embedding=-
1.918)
M2.<a2> = NumberField(x^4 - x^3 - 7*x^2 + 2*x + 9, embedding=-
1.244)
M3.<a3> = NumberField(x^4 - x^3 - 7*x^2 + 2*x + 9,
embedding=1.334)
M4.<a4> = NumberField(x^4 - x^3 - 7*x^2 + 2*x + 9,
embedding=2.827)
```

```
We define coercion maps, to work with these fields.
```

We define coercion maps, to work with these fields.

```
RR.coerce_map_from(M1);
RR.coerce_map_from(M2);
RR.coerce_map_from(M3);
RR.coerce_map_from(M4);
```

```
Composite map:
  From: Number Field in a4 with defining polynomial x^4 - x^3 -
7*x^2 + 2*x + 9
  To:   Real Field with 53 bits of precision
  Defn:   Generic morphism:
          From: Number Field in a4 with defining polynomial x^4
x^3 - 7*x^2 + 2*x + 9
          To:   Real Lazy Field
          Defn: a4 -> 2.827407778337769?
        then
          Conversion via _mpfr_ method map:
          From: Real Lazy Field
          To:   Real Field with 53 bits of precision
```

```
RR(a1)
```

```
-1.91783324472545
```

```
RR(a2)
```

```
-1.24389403467592
```

```
RR(a3)
```

```
1.33431950106360
```

```
RR(a4)
```

```
2.82740777833777
```

```
Below, we define $K$ for use in SAGE.  We find that $K$ has class
number $1$, and discriminant $26569 = 163^2$.
```

Below, we define K for use in SAGE. We find that K has class number 1, and discrim

```
K.<a> = NumberField(x^4 - x^3 - 7*x^2 + 2*x + 9, 'b'); K
```

```
Number Field in a with defining polynomial x^4 - x^3 - 7*x^2 + 2
9
```

```
K.class_group()
```

```
Class group of order 1 with structure  of Number Field in a witl
defining polynomial x^4 - x^3 - 7*x^2 + 2*x + 9
```

```
K.discriminant()
```

```
26569
```

```
sqrt(26569)
```

```
163
```

```
Now comes the computation of local Artin L-functions.
```

Now comes the computation of local Artin L-functions.

```
We make a list of primes $P$, and a list of Frobenius conjugacy
classes in $A4$, by factoring $f$, modulo various primes.
```

We make a list of primes P, and a list of Frobenius conjugacy classes in $A4$, by factori

```
P = pari.primes_up_to_n(2500); P
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59,
67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131,
139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199,
211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277,
281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359,
367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439,
443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521,
523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607,
613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683,
691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773,
787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863,
877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967,
971, 977, 983, 991, 997, 1009, 1013, 1019, 1021, 1031, 1033, 10
1049, 1051, 1061, 1063, 1069, 1087, 1091, 1093, 1097, 1103, 110
1117, 1123, 1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193, 120
1213, 1217, 1223, 1229, 1231, 1237, 1249, 1259, 1277, 1279, 128
1289, 1291, 1297, 1301, 1303, 1307, 1319, 1321, 1327, 1361, 136
1373, 1381, 1399, 1409, 1423, 1427, 1429, 1433, 1439, 1447, 145
1453, 1459, 1471, 1481, 1483, 1487, 1489, 1493, 1499, 1511, 152
1531, 1543, 1549, 1553, 1559, 1567, 1571, 1579, 1583, 1597, 160
1607, 1609, 1613, 1619, 1621, 1627, 1637, 1657, 1663, 1667, 166
1693, 1697, 1699, 1709, 1721, 1723, 1733, 1741, 1747, 1753, 175
1777, 1783, 1787, 1789, 1801, 1811, 1823, 1831, 1847, 1861, 186
1871, 1873, 1877, 1879, 1889, 1901, 1907, 1913, 1931, 1933, 194
1951, 1973, 1979, 1987, 1993, 1997, 1999, 2003, 2011, 2017, 202
2029, 2039, 2053, 2063, 2069, 2081, 2083, 2087, 2089, 2099, 211
2113, 2129, 2131, 2137, 2141, 2143, 2153, 2161, 2179, 2203, 220
2213, 2221, 2237, 2239, 2243, 2251, 2267, 2269, 2273, 2281, 228
2293, 2297, 2309, 2311, 2333, 2339, 2341, 2347, 2351, 2357, 237
2377, 2381, 2383, 2389, 2393, 2399, 2411, 2417, 2423, 2437, 244
2447, 2459, 2467, 2473, 2477]
```

```
f = pari("x^4 - x^3 - 7*x^2 + 2*x + 9");
```

```
The following function figures out which conjugacy class in $A_4$
a given Frobenius element lives in.
```

The following function figures out which conjugacy class in A_4 a given Frobenius elem

```
def FrobCC(p):
    v = 0
    s = gp.get(f.factormod(p, flag=1))
    if s == '[1, 1; 1, 1; 1, 1; 1, 1]':
        v = 1
    if s == '[2, 1; 2, 1]':
        v = 2
```

```
    if s == '[1, 1; 3, 1]':
        v = 3
    return v
```

```
FrobCC(7)
```

3

```
FrobCC(163)
```

0

```
FrobCC(241)
```

1

```
$\omega$ is a primitive cube root of unity.  We view it in a
number field, to speed up computation.
```

ω is a primitive cube root of unity. We view it in a number field, to speed up computati

```
lp = pari("x^2 + x + 1");
```

```
lp.polroots()
```

[-0.500000000000000 - 0.866025403784439*I, -0.500000000000000 + 0.866025403784439*I]~

```
E.<omega> = NumberField(x^2 + x + 1); E
```

Number Field in omega with defining polynomial x^2 + x + 1

```
omega^4
```

omega

```
omega+omega^10
```

2*omega

```
$k$ is the field with $163$ elements.
```

k is the field with 163 elements.

```
k = Integers(163)
```

```
For every $n$, $Cub(n) = 1, \omega, \omega^2$, depending on the
class of $n \in k^\times / k^{\times 3}$.
```

For every n, $Cub(n) = 1, \omega, \omega^2$, depending on the class of $n \in k^\times / k^{\times 3}$.

```
def Cub(n):
```

```
    v = 0
    if k(n^54) == 1:
        v = 1
    if k(n^54) == 104:
        v = omega
    if k(n^54) == 58:
        v = omega^2
    return v
```

```
Chi4(p) computes $x_p^4$, essentially the fourth power of the
centric character of a lift.
```

Chi4(p) computes x_p^4, essentially the fourth power of the centric character of a lift.

```
def Chi4(p):
    v = 0
    FC = FrobCC(p)
    if FC == 1:
        v = Cub(p)^(-1)
    if FC == 2:
        v = Cub(p)^(-2)
    if FC == 3:
        v = Cub(p)^(-3)
    return v
```

```
[Chi4(elem) for elem in P]
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

```
K.factor(163)
```

```
(Fractional ideal (a^3 - 4*a - 4)) * (Fractional ideal (-4*a^3 -
9*a^2 + 16*a - 26))^3
```

```
KI1 = K.ideal(a^3 - 4*a-4); KI1
```

```
Fractional ideal (a^3 - 4*a - 4)
```

```
KI2 = K.ideal(-4*a^3 + 9*a^2 + 16*a - 26); KI2
```

```
Fractional ideal (-4*a^3 + 9*a^2 + 16*a - 26)
```

```
KI1.ramification_index()
```

```
1
```

```
FF1 = KI1.residue_field(); FF1
```

```
Residue field of Fractional ideal (a^3 - 4*a - 4)
```

```
FF2 = KI2.residue_field(); FF2
```

```
Residue field of Fractional ideal (-4*a^3 + 9*a^2 + 16*a - 26)
```

```
QR1 = Set([elem^2 for elem in FF1]); QR1
```

```
{0, 1, 4, 6, 9, 10, 14, 15, 16, 21, 22, 24, 25, 26, 33, 34, 35,
38, 39, 40, 41, 43, 46, 47, 49, 51, 53, 54, 55, 56, 57, 58, 60,
62, 64, 65, 69, 71, 74, 77, 81, 83, 84, 85, 87, 88, 90, 91, 93,
96, 97, 100, 104, 111, 113, 115, 118, 119, 121, 126, 131, 132, 1
134, 135, 136, 140, 143, 144, 145, 146, 150, 151, 152, 155, 156,
158, 160, 161}
```

```
QR2 = Set([elem^2 for elem in FF2]); QR2
```

```
{0, 1, 4, 6, 9, 10, 14, 15, 16, 21, 22, 24, 25, 26, 33, 34, 35,
38, 39, 40, 41, 43, 46, 47, 49, 51, 53, 54, 55, 56, 57, 58, 60,
62, 64, 65, 69, 71, 74, 77, 81, 83, 84, 85, 87, 88, 90, 91, 93,
96, 97, 100, 104, 111, 113, 115, 118, 119, 121, 126, 131, 132, 1
134, 135, 136, 140, 143, 144, 145, 146, 150, 151, 152, 155, 156,
158, 160, 161}
```

```
def QSym1(x):
    S = 1
    xr1 = FF1(x*2)
    if xr1 in QR1:
        S = -1
    if xr1 == 0:
        S = 0
    return S
```

```
def QSym2(x):
    S = 1
    xr2 = FF2(x*2)
    if xr2 in QR2:
```

```
        S = -1
    if xr2 == 0:
        S = 0
    return S
```

```
def QSymb(x):
    S = QSym1(x) * QSym2(x)
    return S
```

```
Below, we pull out a generator of a prime ideal occurring in the
factorization of a prime in $O_K$ (embedded in the reals).
```

Below, we pull out a generator of a prime ideal occurring in the factorization of a prim

```
list(M1.factor(7))[0][0].gens_reduced()[0]
```

```
-a1^3 - a1^2 + 5*a1 + 8
```

```
FrobCC(7)
```

```
3
```

```
Chi3 computes $x_p^3$, essentially the third power of the centric
character of a lift.
```

```
Traceback (click to the left of this block for traceback)
...
SyntaxError: invalid syntax
```

```
K.factor(7)
```

```
(Fractional ideal (-a^3 - a^2 + 5*a + 8)) * (Fractional ideal (-
2))
```

```
M1.factor(7)
```

```
(Fractional ideal (-a1^3 - a1^2 + 5*a1 + 8)) * (Fractional ideal
(-a1 + 2))
```

```
def Chi(p):
    v = 0
    m1 = list(M1.factor(p))[0][0].gens_reduced()[0]
    m2 = list(M2.factor(p))[0][0].gens_reduced()[0]
    m3 = list(M3.factor(p))[0][0].gens_reduced()[0]
    m4 = list(M4.factor(p))[0][0].gens_reduced()[0]
    v = sgn(RR(m1)) * sgn(RR(m2)) * sgn(RR(m3)) * sgn(RR(m4))
    return v
```

```
Chi(163)
```

```
-1
```

```
Chi computes $x_p$, identified as $x_p^3 \cdot x_p^4$, since the
centric character has order $2$ or $6$.
```

Chi computes x_p, identified as $x_p^3 \cdot x_p^4$, since the centric character has order 2 or 6.

```
Update:  We ignore $x_p^4$, since it is trivial!
```

Update: We ignore x_p^4, since it is trivial!

```
def ChiQ(p):
    k1 = list(K.factor(p))[0][0].gens_reduced()[0]
    v = QSymb(k1)
    return v
```

```
ChiList = [Chi(elem) for elem in P]
```

```
ChiList
```

```
[1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1,
-1, 1, 1, -1, -1, 1, 1, 1, -1, 1, -1, -1, -1, -1, 1, -1, 1, 1,
-1, -1, -1, 1, -1, 1, -1, -1, 1, -1, -1, 1, -1, 1, 1, 1, -1, -1,
1, 1, -1, 1, 1, 1, 1, -1, -1, -1, 1, -1, 1, 1, -1, -1, 1, 1, 1,
-1, 1, -1, -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, -1, -1, 1, -1, 1,
-1, 1, -1, -1, -1, 1, -1, -1, 1, -1, 1, 1, -1, -1, 1, -1, -1,
-1, 1, -1, -1, -1, -1, -1, 1, 1, -1, -1, -1, -1, 1, 1, -1, -1,
-1, 1, -1, 1, -1, 1, 1, 1, -1, 1, -1, -1, -1, 1, 1, 1, 1, -1,
-1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
1, -1, -1, 1, -1, -1, -1, -1, 1, 1, -1, -1, -1, 1, 1, -1, -1,
-1, 1, 1, -1, -1, -1, -1, -1, -1, 1, -1, 1, -1, 1, 1, -1, -1, 1,
-1, -1, -1, -1, -1, -1, 1, 1, -1, 1, -1, -1, -1, 1, -1, 1, -1,
-1, -1, -1, -1, 1, -1, -1, 1, 1, -1, -1, 1, 1, 1, -1, -1, 1, -1,
-1, -1, 1, -1, -1, 1, -1, -1, -1, 1, -1, -1, -1, -1, 1, 1, 1,
-1, 1, -1, 1, -1, -1, -1, -1, 1, 1, 1, -1, 1, 1, 1, -1, 1, -1,
-1, -1, 1, -1, 1, 1, -1, -1, 1, -1, -1, 1, -1, 1, 1, -1, -1, 1,
-1, -1, -1, -1, -1, -1, 1, -1, -1, 1, -1, -1, 1, -1, 1, -1, -1,
-1, -1, -1, -1, 1, -1, -1, -1, 1, 1, -1, -1, -1, -1, 1, -1, -1,
1, -1, 1, -1, 1, -1, -1, -1, 1, -1, 1, -1, 1, 1, -1, 1, 1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1]
```

```
len(ChiList)
```

```
367
```

```
sum(ChiList)
```

```
-99
```

```
FrobCC(5)
```

2

```
Chi(5)
```

-1

```
def TD(p):
    t = 0
    FC = FrobCC(p)
    x = Chi(p)
    d = Cub(p)
    if FC == 1:
        t = 2*x
    if FC == 2:
        t = 0
        d = -x
    if FC == 3:
        t = -x/d
    if p == 163:
        t = -1
        d = 0
    return [t,d]
```

```
TD(163)
```

[-1, 0]

```
TDList = Family(P, lambda p:TD(p) ); TDList
```

```
Lazy family (<lambda>(i))_{i in [2, 3, 5, 7, 11, 13, 17, 19,
23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89,
101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163,
167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233,
239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311,
313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389,
397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463,
467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563,
569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641,
643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727,
733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821,
823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907,
911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997,
1009, 1013, 1019, 1021, 1031, 1033, 1039, 1049, 1051, 1061, 1063
1069, 1087, 1091, 1093, 1097, 1103, 1109, 1117, 1123, 1129, 1151
1153, 1163, 1171, 1181, 1187, 1193, 1201, 1213, 1217, 1223, 1229
1231, 1237, 1249, 1259, 1277, 1279, 1283, 1289, 1291, 1297, 1301
1303, 1307, 1319, 1321, 1327, 1361, 1367, 1373, 1381, 1399, 1409
1423, 1427, 1429, 1433, 1439, 1447, 1451, 1453, 1459, 1471, 1481
1483, 1487, 1489, 1493, 1499, 1511, 1523, 1531, 1543, 1549, 1553
1559, 1567, 1571, 1579, 1583, 1597, 1601, 1607, 1609, 1613, 1619
1621, 1627, 1637, 1657, 1663, 1667, 1669, 1693, 1697, 1699, 1709
1721, 1723, 1733, 1741, 1747, 1753, 1759, 1777, 1783, 1787, 1789
1801, 1811, 1823, 1831, 1847, 1861, 1867, 1871, 1873, 1877, 1879
1889, 1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987
1993, 1997, 1999, 2003, 2011, 2017, 2027, 2029, 2039, 2053, 2063
2069, 2081, 2083, 2087, 2089, 2099, 2111, 2113, 2129, 2131, 2137
2141, 2143, 2153, 2161, 2179, 2203, 2207, 2213, 2221, 2237, 2239
2243, 2251, 2267, 2269, 2273, 2281, 2287, 2293, 2297, 2309, 2311
2333, 2339, 2341, 2347, 2351, 2357, 2371, 2377, 2381, 2383, 2389
2393, 2399, 2411, 2417, 2423, 2437, 2441, 2447, 2459, 2467, 2473
2477]}
```

```
TDList[163]
```

```
[-1, 0]
```

```
var('t,d,X')
EF = (1 - t*X + d*X^2)^(-1)
EFT = taylor(EF,X,0,15);
```

```
EFT.coefficient(X^10)
```

```
-9*d*t^8 - d^5 + t^10 + 28*d^2*t^6 - 35*d^3*t^4 + 15*d^4*t^2
```

```
def APP(p,n):
    a = 0
    tdp = TDList[p]
    t = tdp[0]
    d = tdp[1]
```

```
    if n == 0:
        a = 1
    if n == 1:
        a = t
    if n == 2:
        a = t^2 - d
    if n == 3:
        a = t^3 - 2*d*t
    if n == 4:
        a = d^2 + t^4 - 3*d*t^2
    if n == 5:
        a = t^5 - 4*d*t^3 + 3*d^2*t
    if n == 6:
        a = -(d^3 - t^6 + 5*d*t^4 - 6*d^2*t^2)
    if n == 7:
        a = t^7 - 6*d*t^5 + 10*d^2*t^3 - 4*d^3*t
    if n == 8:
        a = d^4 + t^8 - 7*d*t^6 + 15*d^2*t^4 - 10*d^3*t^2
    if n == 9:
        a = t^9 - 8*d*t^7 + 21*d^2*t^5 - 20*d^3*t^3 + 5*d^4*t
    if n == 10:
        a = -(d^5 - t^10 + 9*d*t^8 - 28*d^2*t^6 + 35*d^3*t^4 -
15*d^4*t^2)
    if n == 11:
        a = t^11 - 10*d*t^9 + 36*d^2*t^7 - 56*d^3*t^5 +
35*d^4*t^3 - 6*d^5*t
    if n == 12:
        a = d^6 + t^12 - 11*d*t^10 + 45*d^2*t^8 - 84*d^3*t^6 +
70*d^4*t^4 - 21*d^5*t^2
    if n > 12:
        a = error
    return a
```

```
PP = []
for n in range(1,2001):
    if is_prime_power(n):
        PP = PP + [n]
```

```
def AN(n):
    a = 0
    fac = factor(n)
    applist = [APP(elem[0], elem[1]) for elem in fac]
    a = prod(applist)
    return a
```

```
AN(163)
```

```
-1
```

```
ANList = [AN(elem) for elem in range(1,2400)]
```

```
ANList[160:170]
```

```
[0, 1, -1, 0, 0, omega, omega, -1, -1, 0]
```

```
ANList[163*2 - 1]
```

```
-omega - 1
```

```
old_prec = pari.set_real_precision(60)
```

```
om = pari(e^(2*pi*I/3)); om
```

```
-0.500000000000000000000000000000000000000000000000000000000000
0.866025403784438646763723170752936183471402626905190314027903*I
```

```
def Maass(x,y,m):
    SUM = 0
    SY = pari(sqrt(y))
    TPY = pari(2*pi*y)
    TPX = pari(2*pi*x)
    for n in range(1,m):
        TPNY = pari(TPY*n)
        TPNX = pari(TPX*n)
        ANN = pari(ANList[n-1]).lift()(om)
        SUM = SUM + ANN * SY * pari(0).besselk(TPNY,
precision=128) * pari(TPNX).sin(precision=128)
    return SUM
```

```
ANList
```

```
[1, omega + 1, -omega, 0, 0, 1, -omega - 1, 1, 0, 0, omega, 0, (
-omega, 0, omega + 1, 0, 0, omega, 0, -1, -1, 0, -omega, -1, 0,
0, omega, 0, 0, 0, omega + 1, 0, 0, 0, -1, 0, 0, -omega - 1,
-omega - 1, -omega, 0, 0, 0, -omega - 1, 1, 0, -omega - 1, 0, 0,
omega + 1, 0, -omega - 1, omega + 1, -1, 0, 0, 0, 0, 0, 1, 0, or
omega + 1, 0, 0, 0, -omega, 0, omega, 0, omega, 0, 1, 0, omega,
-omega, -omega, omega + 1, 0, 0, 1, omega + 1, omega, omega + 1,
0, 0, 0, -omega, 0, 0, -omega, 0, 0, 0, -omega - 1, 0, omega + 1
0, 0, -omega - 1, 0, -omega - 1, 0, 0, -omega, -omega - 1, omega
0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, omega + 1, -omega - 1, 0,
-omega, 0, 1, omega, 0, 0, -omega - 1, 0, -omega, 0, -1, 1, 0, (
-1, 0, 0, -omega, -1, omega + 1, omega, 0, omega + 1, 0, 0, 0, -
```

0, 0, 0, 1, -1, 0, 0, omega, omega, -1, -1, 0, 0, 0, omega + 1,
omega, omega + 1, -1, 0, omega, -omega - 1, 0, -omega, 0, 0, 0,
0, 0, 0, -omega - 1, 0, 0, -omega, 0, 1, 0, 0, omega + 1, 0, 0,
1, -omega, 1, 0, 0, omega, 0, 0, -omega - 1, 0, 0, 0, -omega -
-omega, 0, 1, 0, -omega, omega + 1, 0, 0, 0, omega + 1, 0, 0,
-omega, 0, 0, -omega, 0, -omega, omega, -omega, 0, 0, 0, omega
0, omega + 1, 0, -2, 0, 0, 0, 0, -omega - 1, 0, 0, 1, 0, omega,
0, 0, 0, 0, omega, -omega, 0, 0, 0, 1, -omega, omega + 1, 0, ome
1, 1, 0, omega + 1, 0, omega, 0, 0, -omega, -omega, 0, -omega,
0, omega + 1, -omega - 1, -omega, 0, 0, 0, omega, 0, -1, 0, -ome
1, 0, omega + 1, 0, 0, 0, omega, 1, 0, 0, -1, omega, -1, -1, 0,
omega, 0, 1, 0, omega, 0, 0, 0, 0, 0, omega + 1, 0, -omega - 1,
-1, 0, 0, 0, 0, -omega - 1, -1, -omega - 1, omega, 0, -2, 0, 0,
0, -omega - 1, -omega, -omega - 1, -1, 0, 0, 0, -1, -omega, 0,
omega, 2, 0, 0, omega, 0, 0, 0, 0, 0, 0, 0, -omega, omega + 1,
1, 0, 0, 0, 0, omega + 1, 0, 0, 0, 0, 0, omega + 1, 0, 0, -omega
1, 0, -omega, 0, 0, 0, 0, -omega - 1, 1, 0, 0, 0, 0, -omega, 0,
0, -omega - 1, omega, 0, 0, omega, 0, -omega, -omega - 1, omega,
omega + 1, 0, 0, 0, omega + 1, 0, 0, omega + 1, 0, -1, 0, 0, 0,
0, -omega - 1, -omega, -omega, 0, omega, 0, 0, 0, 0, -omega, 0,
0, 0, 0, omega + 1, omega + 1, 0, 0, 0, 0, omega, -omega - 1, 0,
0, omega + 1, 0, 0, omega, -omega - 1, -omega - 1, 0, 0, 1, 0,
0, omega + 1, -omega, 1, 0, 0, 0, 1, -omega - 1, -1, 0, 1, 2, 0,
-omega, 0, 0, 0, omega + 1, omega, -omega, 0, 0, omega, -omega
0, 0, -2*omega - 2, 0, 0, 0, 0, omega + 1, 0, omega, 0, -omega
0, 0, 0, 0, 0, -1, omega + 1, -omega - 1, 0, omega + 1, -1, -ome
0, 0, 0, omega, 0, omega, 0, 1, 1, omega, -1, 0, 0, 1, 0, 1, 0,
omega, 0, -omega - 1, 0, 1, 1, 0, omega, -1, 0, 0, 0, 0, omega
0, omega + 1, -1, omega, 0, 0, -omega, -1, -omega - 1, 0, 0, 0,
0, 0, 1, -omega - 1, 0, 1, 1, 0, 0, omega, 0, 0, 0, 0, omega, om
+ 1, 0, 0, 1, -1, -omega, -omega - 1, 0, -omega, 0, 0, -1, 0, 0,
-omega, -omega - 1, 0, 0, -omega, -omega, 0, omega, 0, omega, 0,
0, 0, 1, 0, 0, -1, 0, 0, 0, 0, 2, omega, omega, -omega - 1, 0,
-omega - 1, -omega - 1, 0, -omega, 0, 0, 0, omega, -1, 0, 1, -om
- 1, omega + 1, omega + 1, 0, 0, -1, -omega, 0, 1, 0, -1, 0, 0,
0, omega, 0, omega, 0, 0, 0, -omega, 0, 0, omega, -omega - 1, -
- 1, 0, 0, 0, -2, -omega, 0, 0, 0, 0, -2, -omega - 1, 0, -omega,
-1, omega + 1, 0, omega + 1, -2*omega - 2, 0, omega + 1, 0, 0,
1, 0, 0, 0, 0, 1, -1, 0, 0, -omega - 1, -1, 0, 0, 0, 0, 0, 0, -
- 1, -omega - 1, 1, 0, 0, omega, 0, 0, 2*omega + 2, 0, omega +
0, -omega - 1, 0, -omega, 0, 0, omega, 0, 0, omega, 0, omega +
0, omega + 1, 0, 0, 0, 0, 1, omega, -omega - 1, 0, -omega, 0,
2*omega, 0, -omega, 0, omega, 0, 1, 0, 0, 0, -omega, omega, 0,
-1, 0, omega + 1, 0, 0, 0, -omega - 1, 0, 0, omega, 0, 0, omega,
2, -omega, omega + 1, 0, 0, 0, 0, 0, 0, 0, omega + 1, 0, omega,
0, -omega, 0, 0, -omega - 1, 0, omega + 1, 0, omega + 1, 0, 0,
-omega, 0, 1, 1, 0, omega + 1, 0, omega, 0, 0, -omega, 2, 0, -om
- 1, 0, omega, 0, 0, -1, 0, 0, -omega, 1, 0, 0, 0, -1, -omega -
0, 0, 0, 1, -omega - 1, 0, 0, -omega, 0, omega + 1, 0, 0, 0, ome

```
1, omega, 0, 0, 0, -omega - 1, 0, omega + 1, -omega - 1, 0, -ome
1, 0, -omega, 0, -omega - 1, 0, 0, -omega, 0, 0, 0, 1, -omega, (
-1, 1, 0, 0, 0, 0, 0, -omega - 1, 0, 0, 0, 2, 0, 0, -omega - 1,
omega + 1, 0, -omega - 1, 0, omega + 1, 0, 0, 0, 0, omega, omega
-omega - 1, 0, 0, -omega - 1, 0, 0, 0, 0, -omega - 1, -omega, 1,
-omega, 0, omega, 0, 0, omega, -omega - 1, 0, 0, 0, omega + 1, (
-omega, 0, -omega, 0, 0, 0, 0, 0, omega + 1, omega, -omega - 1,
omega + 1, omega, 0, 0, 0, -omega - 1, omega, -1, 1, 0, 0, -1, (
-2, 0, omega + 1, 0, 0, 0, 0, -omega, 0, 0, omega, 0, 0, 0, omeg
1, 2*omega + 2, 0, 0, -omega - 1, 1, 0, 0, 0, 0, 0, 0, 0, omega,
0, 0, 1, 1, 0, -2, 0, 0, 0, -1, -omega, omega, 0, -1, 0, 0, 0, (
omega, 0, 0, 0, -omega - 1, 0, -1, omega, 0, 0, 0, -1, -1, 0, 0,
-omega, 0, -1, 0, 0, omega + 1, 0, 0, 0, 0, 0, 2*omega, -omega -
0, 0, -omega, -omega, 0, 0, 0, omega, 0, 0, 0, 1, 0, 0, 0, 0, -c
- 1, 0, -omega - 1, -1, 0, 0, 0, -1, omega + 1, 0, omega, omega
0, omega + 1, omega + 1, -1, 0, 0, omega, 0, 0, -omega - 1, omeg
omega + 1, 0, 0, 0, omega + 1, 0, 0, -2*omega, -1, -1, 0, 0, -or
0, -omega, -omega, omega + 1, -omega, 0, 0, 0, 0, 0, -omega, -or
- 1, 0, 0, -omega - 1, 0, 0, 1, 0, 0, omega + 1, 0, -omega - 1,
0, omega, 0, -omega - 1, omega, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
-omega, omega + 1, 0, 0, 0, omega + 1, 0, -2, 0, 0, -omega - 1,
-omega - 1, 0, 0, 0, 1, -omega, 0, 0, 0, omega + 1, -omega - 1,
omega, 0, 1, -omega, 0, -1, 0, 0, -omega, 0, 1, 0, 0, 0, -omega,
0, omega, 0, -1, -omega - 1, 0, 0, 0, -1, -omega - 1, 0, 1, 0,
-omega, 0, 0, omega + 1, 1, 0, 0, 0, 0, 0, 0, -1, 0, -omega - 1,
-omega - 1, 1, 0, 0, 0, 0, 0, 0, -omega, 1, 0, 0, 0, 0, -omega -
-1, 1, 0, 0, 0, 0, 0, omega + 1, 0, 1, 0, 0, 0, 0, omega + 1, or
+ 1, 0, 0, 0, omega + 1, 0, 1, 0, omega + 1, -omega, omega, 0, (
0, 2*omega + 2, 1, -1, omega + 1, -1, omega + 1, 0, 0, 0, 0, ome
1, 0, 0, -omega, 0, -omega - 1, -omega, 0, omega, omega + 1, 1,
0, 0, 0, -omega, 0, 0, -1, 1, 0, omega, 0, omega, omega + 1, 0,
-omega, 0, 0, omega + 1, omega, 0, 0, 0, 0, 1, 0, 0, 1, omega +
0, -omega, omega + 1, 0, 0, omega, -omega - 1, 0, 0, -omega - 1,
-omega - 1, 0, 0, 0, omega + 1, -1, 0, 0, -1, 0, -omega - 1, 0,
0, -1, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -omega, 0, 0, omega,
0, 0, 0, -2*omega - 2, 0, 1, omega + 1, 0, 1, 0, -omega, 0, 0, -
0, -2*omega - 2, -omega, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, omega,
omega, 0, 0, omega, 0, 0, 0, 0, 0, omega, 1, 0, -1, 0, 0, 0, 0,
omega, 0, omega + 1, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, omega + 1,
-omega - 1, 0, -1, -omega, 0, 0, 0, 0, -omega - 1, 0, 0, omega -
0, 1, 0, 0, 0, omega, 0, -1, 0, -omega - 1, 0, -omega, -omega, (
0, 0, -omega, 0, 0, -1, 0, omega + 1, 0, 0, -omega - 1, 0, -1, (
omega, 0, 0, 0, 0, 0, -omega, -omega, omega + 1, -2*omega, 1, 0,
0, 0, -omega - 1, -1, 0, 0, 0, 0, 0, -1, 0, 0, 0, omega, 1, 0, (
-omega - 1, omega, -omega - 1, 0, 0, 0, 0, 0, 0, -omega - 1, ome
1, omega + 1, 0, 0, -1, -omega, 0, 0, omega + 1, 1, 0, 0, 0, -2,
omega, -omega, 0, 1, omega + 1, 0, -omega, -1, 0, 0, 0, omega +
0, 0, 1, 0, omega, 0, 0, 1, 0, 0, 0, 0, omega, 0, -1, -omega - 1
0, 0, omega, 0, 0, -omega, 0, omega + 1, 0, 0, -omega, -omega, (
0, omega, 0, -omega - 1, 0, 0, 0, -1, -1, -omega, 0, -omega - 1,
```

0, omega, 0, -omega - 1, 0, 0, 0, -1, -1, -omega, 0, -omega - 1,
2*omega + 2, 0, 0, 0, omega, 1, 0, -omega - 1, 0, -omega - 1, -c
- 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, omega, -omega - 1, 0, 0, -1, ome
1, 0, omega + 1, 0, 0, 0, -omega, 0, 0, -omega, 0, -omega, omega
1, 0, 0, omega, -omega - 1, 0, 0, omega, 0, 0, -omega, 0, -omega
0, 0, 0, 0, 0, omega + 1, omega, 0, -omega, omega, omega + 1, 0,
-1, -omega, 0, -1, 0, 0, 0, 0, 2*omega + 2, 0, omega + 1, -1,
-omega, omega, 0, 0, -1, -omega - 1, 0, 0, 0, omega, 0, 0, 0, 0,
0, 1, 0, 0, omega + 1, 0, 0, -1, omega, 0, -1, 0, 0, -omega, 0,
0, 0, 0, 0, 0, omega + 1, 0, -omega, 0, 0, -omega, 0, omega + 1,
-omega - 1, 1, 0, omega, -omega, 0, 0, 0, -1, 0, 0, omega, 0, 0,
-omega - 1, 0, omega + 1, 0, 0, 0, 0, 0, 0, 0, 0, omega, 0, -ome
0, 0, -1, -omega, 0, 0, 0, 1, -omega, 0, -1, -omega, omega + 1,
0, 0, 0, 0, -omega, 0, omega + 1, -omega - 1, -omega, 0, -omega
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, omega + 1, 2*omega + 2, 0, 1, 0, -
0, omega, 0, 0, 0, -omega - 1, -omega, omega, 0, 0, 0, 0, -omega
1, 0, 2*omega + 2, -1, 0, omega, 0, 0, -omega, -omega - 1, omega
0, 0, -omega, 0, 0, omega + 1, omega, -omega, 0, 0, 0, 0, 1, 0,
-omega - 1, 0, omega + 1, -1, 0, 0, 0, -omega, 0, 0, omega, 0,
-omega - 1, -omega, 0, 0, omega + 1, 0, 0, 0, 0, omega + 1, 2,
-omega, 0, 0, 1, omega + 1, omega + 1, 0, 0, 1, -1, 0, 0, -1, 0,
0, 0, 0, 0, 0, -omega, omega, 0, -omega - 1, 0, 0, 0, 0, omega,
omega + 1, omega + 1, 0, omega + 1, 0, 0, omega, 0, 0, 0, -omega
0, 0, -2*omega, 0, 1, 0, 0, 0, omega + 1, 0, 0, -1, 0, -omega, c
+ 1, 0, omega, 0, 0, -1, 0, 0, 0, 0, 0, 0, -1, -omega, 2, 0, -or
-omega - 1, 0, 0, 0, 0, -2, -omega, 0, -omega - 1, 0, 0, -omega
-2*omega - 2, omega + 1, 0, -1, omega, omega + 1, 0, 0, 0, omega
1, -omega, 0, 0, -1, 0, 0, 0, 0, omega, 1, -1, -omega, 0, -omega
0, -omega, 0, omega, -omega - 1, 0, -omega - 1, 0, 0, 0, omega -
-omega, -omega, 0, omega, 0, 0, 0, 0, 0, -omega, 0, 0, 0, 0, 0,
omega + 1, 0, 0, 0, 0, 0, 0, omega + 1, 1, 0, 0, 0, omega, omega
1, -1, 0, 0, -2*omega - 2, -omega, 0, 0, 0, 0, omega + 1, -omega
-omega - 1, 0, 0, -omega, -1, 1, 0, 0, -omega - 1, omega + 1, 0,
0, omega, -2, -1, 0, 0, 0, 0, -1, 0, 0, 0, 0, -1, 0, 2*omega,
-omega, -1, 0, 0, -omega - 1, 0, 0, omega + 1, 0, 0, 0, 0, 0, 0,
-1, -omega - 1, 2*omega, 0, 0, 0, 0, 0, 0, 0, -omega, -omega - 1
0, omega, 0, 0, omega, -omega, 0, 1, 0, -2, 0, -1, 0, 1, 0, 0, -
-2, 0, 0, 0, omega + 1, 1, 0, 1, 0, 0, -omega - 1, 0, omega, 0,
0, omega + 1, 0, 0, 0, 0, omega, 0, 0, omega, 0, 0, 0, 0, 0, 0,
0, 0, -1, -omega, omega + 1, 0, omega, -omega, -omega, 0, omega,
0, 0, 1, 0, 0, 0, omega, omega, -omega - 1, 0, 0, -1, 0, 0, 0, 0
0, 0, omega, -omega, 0, -omega, 0, 0, omega, 0, -1, omega + 1, 0
0, -omega - 1, -omega, 0, -1, 0, 0, omega + 1, 0, 1, 0, omega +
0, -omega - 1, 0, 0, 0, 1, 0, 0, 2, 0, 0, 0, -omega - 1, omega,
0, 0, 0, 0, 0, 0, 0, 1, 0, 1, omega, 0, 0, 1, -omega - 1, -omega
0, 0, 0, 0, 0, 0, omega + 1, 0, 1, 0, 0, -omega - 1, 0, 0, 0, or
+ 1, -omega, 0, 0, 0, 0, 1, omega + 1, omega + 1, 0, -omega - 1,
omega, omega, 0, 1, omega, -omega, 0, 0, 0, 0, -omega, 0, 0, 0,
0, 1, -1, 0, omega + 1, omega + 1, omega + 1, 0, 0, -1, 0, 0, 0,
-omega - 1, 0, -omega - 1, 0, 0, 0, -1, omega, 0, 0, 0, 0, 0, or

```
-omega - 1, 0, 1, 0, -omega - 1, 0, omega + 1, omega, 0, 0, 0,
-2*omega - 2, -omega, 0, 0, 0, 0, -omega, 0, -omega, 0, 0, 0, 0,
-omega - 1, -omega, 0, omega + 1, omega, 0, 0, 0, omega, 0, 0, (
omega, 0, -omega, -omega, 0, -omega, 1, -1, -omega, 0, 0, omega
0, 1, -omega - 1, 0, 0, 0, -1, 0, 0, 0, omega, 1, 0, 0, -omega -
omega + 1, -omega, 0, omega + 1, 0, -omega, 0, 0, 1, omega + 1,
omega + 1, 0, 0, -2, 0, -2*omega, 0, 0, -omega - 1, 0, -omega, (
0, 0, 0, -omega, 0, -omega - 1, -omega, 0, 0, 0, 0, 0, omega + 1
0, 0, 0, 0, 0, 0, 0, omega, 1, 0, 0, 0, 0, 0, omega + 1, 0, -ome
1, 0, omega, 0, 0, omega, -omega, -omega - 1, 0, 0, 0, -omega, (
0, -omega, -1, 0, 0, 0, -omega, -1, 0, 0, 0, 0, 2*omega + 2, 0,
0, 0, 1, 0, 0, 0, 0, 0, -omega - 1, 0, 0, 0, 0, omega, -omega, (
-omega, omega + 1, omega + 1, 0, 2, 0, 1, omega + 1, 0, 0, 0, 0,
omega + 1, omega, -omega, 0, 0, omega + 1, 0, 0, -omega, 0, -1,
-2*omega, 0, 0, 0, 0, omega, 0, 0, 0, 0, -omega - 1, 0, omega +
omega, 0, omega, -omega, omega + 1, 0, 0, 0, omega, -omega - 1,
0, -1, 0, 0, -omega - 1, 0, -omega - 1, 0, omega, 0, 0, 0, 0, or
+ 1, -omega - 1]
```

```
len(ANList)
```

```
2399
```

```
def Moeb(z):
    w = (z)/((163*z)+1)
    return w
```

```
CC(1/163)
```

```
0.00613496932515337
```

```
CC(1/100 + I/170)
```

```
0.0100000000000000 + 0.00588235294117647*I
```

```
CC(1/110 + I/190)
```

```
0.00909090909090909 + 0.00526315789473684*I
```

```
CC(Moeb((-1/100) + (I / 170)))
```

```
0.00907138149619040 + 0.00446904933593870*I
```

```
Maass(-1/100,1/170,1000)
```

```
-0.021020228494251239546558653805386538153870902756952441901914
0.007989071046714959932605203203894365295079016926385726306112
```

```
Maass(-1/100,1/170,1500)
```

```
-0.021020228494251237899909575506366554652509333170733591877275
0.007989071046714962183150992385917791207027918856989935189562
```

```
Maass(34507/3803941, 17000/3803941, 2000)
```

```
-0.0210202284942512378999095699096593513581660770392626888897217
0.00798907104671496218315099434334069076386056895458915391135600
```

```
Maass(34507/3803941, 17000/3803941, 2400)
```

```
-0.0210202284942512378999095708425699106683617213328060906787933
0.00798907104671496218315098503978026003919984223190634266887274
```

```
def Smoo(x):
    S = CC(x^2 * e^(-x))
    return S
```

```
def STest(X,m):
    R2 = range(1,m)
    SUML = [ANList[elem]*Smoo(elem/X) for elem in R2]
    SUM = sum(SUML)
    return SUM
```

```
def SComp(X):
    S = 164*( (163*13/4) / (42.88*X) )^2
    return S
```

```
STest(13,750)
```

```
-0.875338201196175 - 0.0300324760104438*I
```

```
STest(13,800)
```

```
-0.875338201196175 - 0.0300324760104438*I
```

```
SComp(13)
```

```
148.111752556597
```

```
SComp(200)
```

```
0.625772154551621
```