```
1
2  function IdealConjugate(I, K)
3  // Input: Z_K-Ideal I; Field K
4
5  // Output: Z_K-Ideal which is the complex conugate of I
6
7      gens := [];
8      for g in Generators(I) do
9          Append(~gens, ComplexConjugate(K ! g));
10     end for;
11
12     return ideal<Integers(K)|gens>;
13
14 end function;
15
16
17 function ReduceByIsometry(Lattices)
18 // Input: List of lattices
19
20 // Output: Reduced list for which the elements are
   pairwise non-isometric
21
22     LatticesReduced := [* *];
23     Minima := [* *];
24     NumShortest := AssociativeArray();
25     SizeAuto := AssociativeArray();
26
27
28     for i in [1..#Lattices] do
29         L := Lattices[i];
30
31         min_computed := false;
32         minimum := 0;
33
34         shortest_computed := false;
35         shortest := 0;
36
37         auto_computed := false;
38         auto := 0;
39
40         for j in [1..#LatticesReduced] do
41             M := LatticesReduced[j];
42
43             if not min_computed then
44                 min_computed := true;
45                 minimum := Min(L);
46             end if;
```

```
47
48              if not IsDefined(Minima, j) then
49                  Minima[j] := Min(M);
50              end if;
51
52              if minimum ne Minima[j] then
53                  continue;
54              end if;
55
56
57              if not shortest_computed then
58                  shortest_computed := true;
59                  shortest := #ShortestVectors(L);
60              end if;
61
62              if not IsDefined(NumShortest, j) then
63                  NumShortest[j] := #ShortestVectors(M);
64              end if;
65
66              if shortest ne NumShortest[j] then
67                  continue;
68              end if;
69
70
71              if not auto_computed then
72                  auto_computed := true;
73                  auto := #AutomorphismGroup(L);
74              end if;
75
76              if not IsDefined(SizeAuto, j) then
77                  SizeAuto[j] := #AutomorphismGroup(M);
78              end if;
79
80              if auto ne SizeAuto[j] then
81                  continue;
82              end if;
83
84
85              if IsIsometric(L, M) then
86                  continue i;
87              end if;
88          end for;
89
90          Append(~LatticesReduced, Lattices[i]);
91
92          NewIndex := #LatticesReduced;
93          if min_computed then
```

```
            Minima[NewIndex] := minimum;
        end if;

        if shortest_computed then
            NumShortest[NewIndex] := shortest;
        end if;

        if auto_computed then
            SizeAuto[NewIndex] := auto;
        end if;

    end for;

    return LatticesReduced;
end function;


function ExtremalMinimum(l, n)
// Input: Square-free l in N; n in N

// Output: Minimum that a l-modular lattice of
dimension n must have at least

    if l eq 1 then k := 24;
    elif l eq 2 then k := 16;
    elif l eq 3 then k := 12;
    elif l eq 5 then k := 8;
    elif l eq 6 then k := 8;
    elif l eq 7 then k := 6;
    elif l eq 11 then k := 4;
    elif l eq 14 then k := 4;
    elif l eq 15 then k := 4;
    elif l eq 23 then k := 2;
    end if;

    return 2 + 2*Floor(n/k);
end function;
```