The Baby Step, Giant Step Method

The following prime is not so easy to deal with.

```
p:=633073699
```

633073699

```
alpha:=numlib::primroot(p)
```

3

We first find N for the "giant steps".

```
N:=ceil(sqrt(p-1))
```

25161

Say we want to find L(15679625). We need to generate two lists.

```
b:=15679625:
```

```
firstlist :=
matrix(array(1..N, [powermod(alpha, j, p) $ j=0..N-1])):
```

Now we compute the size of the "giant steps".

```
c:=powermod(alpha, -N, p):
```

```
secondlist :=
matrix(array(1..N, [_mod(b*powermod(c, j, p),p) $ j=0..N-1])):
```

Now we look for a coincidence between the two lists. The obvious method takes N^2 comparisons, which is slow. So let's do something that takes only a single loop.
The following is the concatenation of the two lists:

```
u:=coerce(firstlist,DOM_LIST).coerce(secondlist,DOM_LIST):
```

Now we search for duplications.

```
v:=listlib::removeDuplicates(u, KeepOrder):
```

Now we see how lists u and v differ.

```
for j from 1 to 2*N do
if v[j] <> u[j] then print(j); break; end_if; end_for
```

48697

This was the only slow step; it took 117 seconds of CPU time. OK, that means that at the entry number 48697 of list u, there was a duplication of an earlier entry. So this happened at entry number 48697-N of the second list.

```
j:=48697-N
```

23536

```
t:=secondlist[23536]:
for k from 1 to N do
if firstlist[k] = t then print(k); break; end_if; end_for
```

12815

We can assemble this to get the discrete log, x.

```
x:=12814+23535*N
```

592176949

Check:

```
powermod(alpha, x, p)
```

15679625

```
%=b
```

$$15679625 = 15679625$$

Yep, it works!