// The output are the standard generators of cyclic codes of given n and q.

function baseq\_rep(n,q,irrNum)

listPowers:=[];

for i in [1..irrNum] do

r := n mod q;

Append(~listPowers, r);

n := n div q;

end for;

return listPowers;

end function;

// this function takes a list of powers and the factorization list of lists

// the output is the corresponding polynomial

function vectorToPoly(vecPower,vecFac)

poly:=1;

for i in [1..#vecPower] do

poly:=poly\*vecFac[i]^vecPower[i];

end for;

return poly;

end function;

// this procedure compares with the parameters with the BKLC parameters and write parameters and results to a file F

procedure compareWithBinaryLinear(code,length,generator,degreeG,degreeH,MLW,F,StartT)

LinearGrayMapImage:=false;

if (HasLinearGrayMapImage(code)) then

LinearGrayMapImage:=true;

PrintFile(F, "Linear Gray Map Image");

else

PrintFile(F, "Nonlinear Gray Map Image");

end if;

PrintFile(F, "&");

PrintFile(F, degreeG); // k1

PrintFile(F, "&");

PrintFile(F, degreeH); // k2

PrintFile(F, "&");

PrintFile(F, MLW);

PrintFile(F, "&");

PrintFile(F, generator);

PrintFile(F, "&");

if (LinearGrayMapImage) then

if MLW ge BKLCLowerBound(GF(2), length\*2, degreeG\*2+degreeH) then

if MLW eq BKLCLowerBound(GF(2), length\*2, degreeG\*2+degreeH) then

PrintFile(F, "Good Code");

PrintFile(F, "&");

else

PrintFile(F, "Great Code");

PrintFile(F, "&");

end if;

end if;

else

if MLW ge BKLCLowerBound(GF(2), length\*2, degreeG\*2+degreeH) then

if MLW eq BKLCLowerBound(GF(2), length\*2, degreeG\*2+degreeH) then

PrintFile(F, "Decent Code");

PrintFile(F, "&");

else

if MLW gt BKLCUpperBound(GF(2), length\*2, degreeG\*2+degreeH) then

PrintFile(F, "Very Good Code");

PrintFile(F, "&");

else

PrintFile(F, "Good Code");

PrintFile(F, "&");

end if;

end if;

end if;

end if;

PrintFile(F, Cputime()-StartT);

PrintFile(F, "&&&");

end procedure;

Z4:=IntegerRing(4);

P4<x>:=PolynomialRing(Z4);

Fac:=CyclotomicFactors(Z4, n);

file:="/home/onta1/non\_free\_Z4\_2/Output\_nf/NonFreeCyclicCodesLen" cat IntegerToString(n) cat ".txt";

PrintFile(file, "&&&");

TimeLimit:=10;

skippedTime:=0;

skippedDimCap:=0;

for num in [0..((3^#Fac)-1)] do

startTime:=Cputime();

listPower:=baseq\_rep(num,3,#Fac);

f:=P4!1;

g:=P4!1;

h:=P4!1;

gen:=P4!1;

freeCode:=0;

nonFreeCode:=0;

for i in [1..#Fac] do

if listPower[i] eq 0 then

f\*:=Fac[i];

elif listPower[i] eq 1 then

g\*:=Fac[i];

elif listPower[i] eq 2 then

h\*:=Fac[i];

end if;

end for;

gen:=f\*h+2\*f;

rem:=((x^n)-1) mod gen;

if rem eq 0 then

// print ("rem is zero");

// C:=CyclicCode(n, gen);

// C:Minimal;

freeCode+:=1;

else

C:=CyclicCode(n, gen);

// C:Minimal;

k1:=Degree(g);

k2:=Degree(h);

nonFreeCode+:=1;

if n gt 61 then

if (Minimum(2\*n-2\*k1-k2,2\*k1+k2) le 60) then

MinLeeW:=MinimumLeeWeight(C: MaximumTime:=TimeLimit);

if MinLeeW eq -1 then

skippedTime +:= 1;

end if;

compareWithBinaryLinear(C,n,gen,k1,k2,MinLeeW,file,startTime);

else

skippedDimCap +:= 1;

end if;

else

MinLeeW:=MinimumLeeWeight(C); // no time limit in this case when n is smaller or equal to 61

compareWithBinaryLinear(C,n,gen,k1,k2,MinLeeW,file,startTime);

end if;

end if;

end for;

print("number of free codes"),freeCode;

print("number of non-free codes"),nonFreeCode;

print("number of codes"),freeCode+nonFreeCode;

print("Number of codes skipped due to time limit:"), skippedTime;

print("Number of codes skipped due to the limit of minimum between dimension and n-dimension:"), skippedDimCap;