Question 7

The example version of the Chinese Remainder Theorem has several inefficiencies. Observe that in the Chinese Remainder Theorem the first step is to initialize the M array, where the value of M[i] is the product of all the moduli except moduli[i]. This is performed at the beginning of every function call, which is somewhat inefficient, because it could just be done once, for a single set of moduli. Furthermore, the output of this function is larger than it needs to be, indeed, it need be no larger than the product of all the moduli. In this question, do not merely call built in Sage functions.

1. Write a function to pre-compute the M array, it should also compute the product of all the moduli.
2. Write a version of the CRT function that takes the precomputed M array and a list of residues. Make sure that the output of this function is no larger than it needs to be.

Question 7

The example version of the Chinese Remainder Theorem has several inefficiencies. Observe that in the Chinese Remainder Theorem the first step is to initialize the M array, where the value of M[i] is the product of all the moduli except moduli[i]. This is performed at the beginning of every function call, which is somewhat inefficient, because it could just be done once, for a single set of moduli. Furthermore, the output of this function is larger than it needs to be, indeed, it need be no larger than the product of all the moduli. In this question, do not merely call built in Sage functions.

1. Write a function to pre-compute the M array, it should also compute the product of all the moduli.

CRT\_precompute(moduli):

M = prod(moduli)

Marray = [0 for j in xrange(len(moduli))]

for j in xrange(len(moduli)):

Mpr = ZZ(M/moduli[j])

Mpr\_inv = xgcd(Mpr, moduli[j])[1]

Marray[j] = Mpr\*Mpr\_inv % M

return (Marray, M)

1. Write a version of the CRT function that takes the precomputed M array and a list of residues. Make sure that the output of this function is no larger than it needs to be.

CRT\_reconstruct\_from\_precomp(precomp\_array, residues, M):

X = 0

for j in xrange(len(precomp\_array)):

X += precomp\_array[j]\*residues[j]

X = X % M

return X