This module provides an implementation of the constructive algorithm for $x$ in the Chinese remainder theorem.

```
module CRT ( crt ) where
```

# Helpers

The extended Euclidean algorithm finds the coefficients of Bézout's identity. For two inputs $a$ and $b$ it finds $x$ and $y$ s.t.

$$ax + by = gcd(a, b) \tag{1}$$

We can implement this as follows:

```
extendedEu :: Integer -> Integer -> (Integer, Integer)
extendedEu _ 0 = (1, 0)
extendedEu a b =
  let
    (q, r) = quotRem a b
    (s, t) = extendedEu b r
  in
    (t, s - q * t)
```

As such, we then trivially have:

```
gcd :: Integer -> Integer -> Integer
gcd a b = a*x + b*y
  where (x, y) = extendedEu a b
```

# CRT

## Description

Chinese remainder theorem tells us that for a list of integers $n_1, ..., n_k$ and a second list $a_1, ..., a_k$, there exists soem $x$ that solves the following:

$$\begin{aligned}
x &= a_1 (\text{mod } n_1) \\
x &= a_2 (\text{mod } n_2) \\
&... \\
x &= a_k (\text{mod } n_k)
\end{aligned}$$

And that moreover, all solutions $x$ are congruent modulo $N = \Pi_i n_i$.

## Algorithm

We can generate that $x$ using the following algorithm. First, calculate $N$. Then, for each $i$ we can find $r_i$ and $s_i$ s.t.

$$r_i n_i + \frac{s_i N}{n_i} = 1 \tag{2}$$

we do this by using the extended Euclidean algoirthm on $n_i$ and $N/n_i$. In which case $s_i$ is the second element of the output of `extendedEu`.

We then have a solution $x$ given by

$$x = \sum_{i=1}^{k} \frac{a_i s_i N}{n_i} \tag{3}$$

And we can find the minimal solution by taking mod $N$.

## Implementation

We can implement that as follows.

First, a helper to calculate each ith element of the above sum.

```
getElement:: Integer -> (Integer, Integer) -> Integer
getElement bigN (a, n) = (a*s*bigN) `div` n
  where s = snd $ extendedEu n (bigN `div` n)
```

Then the real thing:

```
crt:: [(Integer ,Integer)] -> Integer
crt cs = sum (map (getElement n) cs) `mod` n
  where n = product $ map snd cs
```