Code Jam 2010 - Qualification Round

Analysis: Fair Warning

This turned out to be the hardest problem in the qualification round. One of the reasons may have been that the solution involves big arithmetic precision. Using big numbers in timed programming contests is sometimes not considered fair because some languages like python or java have internal libraries that deal with this while for other languages you may have needed to write your own or search for a external one like the GNU Multi-Precision Library. Considering that the qualification round was 24 hours long we took this chance to give you a warning and one which is fair ... that big numbers are fair game for now on, so have a library on hand!

This problem talks about divisors and multiples so it hints at using the greatest common divisor concept in some way. To solve the problem we need to find T and afterwards we can easily find y.

Let's simplify the problem and just look at two numbers a and b. In this case we need to find the largest T so that some positive y exists where $a + y$ and $b + y$ are multiples of T. So $(a + y) \bmod T = (b + y) \bmod T$ which means that $(a - b) \bmod T = 0$. Thus T must be a divisor of $|a - b|$.

Coming back to the problem with N numbers, we have proved that T must divide every $|t_i - t_j|$. This means that the solution is the greatest common divisor of all $|t_i - t_j|$ numbers. A simple observation can improve our algorithm from $O(N^2)$ to $O(N)$ iterations of the Euclidean algorithm. Let's say $a \le b \le c$, then $gcd(b - a, c - a) = gcd(b - a, c - a, c - b)$. To prove this we look at the first iteration of the Euclidean algorithm. Since $a \le b \le c$ it means that $b - a \le c - a$ so in the first step we subtract $b - a$ from $c - a$: $gcd(b - a, c - a) = gcd(c - b, b - a)$, this proves our observation. Now we can just compute the greatest common divisor of all $|t_i - t_1|$ to find T.

Here's some python code from Xiaomin Chen that solves one test case:

```
def Gcd(a, b):
  if b == 0:
    return a
  return Gcd(b, a % b)

def Solve(L):
  y = L[0]
  L1 = [abs(x - y) for x in L]
  g = reduce(Gcd, L1)
  if y % g == 0:
    return 0
  else:
    return g - (y % g)
```

Useful concepts: Greatest common divisor, Euclidean algorithm, Arbitrary precision arithmetic.