1 The problem

This is a problem 145 from Project Euler.

2 Definitions

```
import Data.Char (digitToInt)
```

Let's just start with brute-force solution and optimize it until we don't get good enough perfomance:

```
max_n = 1000000000


isReversible :: Int → Bool
isReversible n = last (show n) ≢ '0' ∧ all odd ds
  where ds = digits (n + revr n)


main :: IO ()
main = let rvs = filter isReversible [1 .. max_n]
  in do print $ length rvs
```

With $max\_n = 1000000$ it takes 25s to run, so I assume it will run for $25 * 1000 = 25000 seconds \approx 7 hours$ with $max\_n = 1000000000$.

3 Optimizations

Profiling shows that *digits* is a bit more costly than *revr*. Let's optimize it a little bit:

```
digits :: Int → [Int]
digits n = map digitToInt $ show n
```

Now 1000000 n's are tested for 8 seconds, that's 3 times better.

The most time-consuming function now is *revr*. I have no idea how to optimize it. We can try to get rid of it later. But now let's try to avoid calling it at all:

```
isReversible' :: Int → Bool
isReversible' n = ¬ leading_zeros ∧ diff_mod ∧ one_if_overflow ∧ all_odd
  where leading_zeros = last_digit ≡ 0
    all_odd = all odd ds
    ds = digits (n + revr n)
    revr n = read rsh
    sh =  show n
```

```
    rsh = reverse sh
    last_digit = n `mod` 10
    first_digit = n `div` powerOfTen
    powerOfTen = (head $ filter (>n) powersOfTen) `div` 10
    diff_mod = first_digit `mod` 2 ≢ last_digit `mod` 2
    one_if_overflow = (last_digit + first_digit < 10) ∨ (last_digit `mod` 2 ≡ 1)


powersOfTen = map (10↑) [1..]
```

After adding a *diff_mod* check: 5 seconds.

After adding a check about *one_if_overflow*: 3 seconds.

It will consume like 12 minutes if I parallelize it on my four cores. Let's try another approach to reversing, and test it as we go:

```
reverse_and_test :: Int → Bool
reverse_and_test n = nm10 ≢ 0 ∧ odd (digitToInt (head sn) + nm10) ∧ every_odd sn rd 0
  where rd = reverse sn
    sn = show n
    nm10 = n `mod` 10
    every_odd (s : sn) (r : rd) ost = odd su ∧ every_odd sn rd ost'
      where su = digitToInt s + digitToInt r + ost
        ost' = su `div` 10
    every_odd [] _ _ = True


isReversible = reverse_and_test
```

Oh, that is *really* better: 0.5 seconds for 1 million n's. But it's going to take 10 minutes for 1 billion, let's optimize further.

Actually I didn't optimize it further, ah, whatever.