Problem 36

Find the sum of all numbers less than one million, which are palindromic in base 10 and base 2.

In the overview for problem 4 a function was presented that checks if a number is a palindrome in base 10. This can be extended to a function that does the job for any specified base:

```
Function isPalindrome(n,base)
     reversed := 0
     k := n
     while k > 0
          reversed := base*reversed + k mod base
          k := k div base
     return (n = reversed)
```

It is easy to see that a number in base 2 must be odd to be a palindrome.

So our basic program might run as follows:

```
i := 1
sum :=  0
while i < 1000000
     if isPalindrome(i,10) and isPalindrome(i,2) then
          sum := sum + i
     i := i + 2
output sum
```

This works for the given limit but becomes rather slow for higher targets. However, it is possible to generate palindromes in any given base rather than checking all numbers for two bases.

Suppose we have a palindrome of the form *xyzzyx* in base *b*, then the first 3 digits define the palindrome. However, the 3 digits *xyz* also define the palindrome *xyzyx*. So the 3-digit number *xyz* defines a 5-digit palindrome and a 6 digit palindrome. From which follows that every positive number less than b^n generates two palindromes less than b^{2n}. This holds for every base *b*.

For the given limit it follows that we need only generate about 2000 palindromes in one of the two bases and check it in the other. This is much faster than checking 1000000 (or 500000) numbers for one of the two bases first. For a limit of 10^8 we would need to generate only about 20000 palindromes instead of checking $5 \cdot 10^7$ numbers.

Copyright Project Euler, further distribution without the consent of the author(s) prohibited
Author: hk

Let's define our general palindrome generator to have a choice between an odd-digit palindrome and an even-digit palindrome.

Our palindrome generator might then look like:

```
Function makePalindrome(n,base,oddlength)
res := n
if oddlength then n := n div base
while n > 0
     res := base*res + n mod base
     n := n div base
return res
```

To generate palindromes in base 2 this function can be made more efficient using 'shift' and 'and' operations rather than 'mod' and 'div' operators like this:

```
Function makePalindromeBase2(n,oddlength)
res := n
if oddlength then n := n shiftRight 1
while n>0
     res := res shiftLeft 1 + n and 1
     n := n shiftRight 1
return res
```

Our program might then look like:

```
limit := 1000000
sum := 0
i := 1
p := makePalindromeBase2(i,true)
while p < limit
     if isPalindrome(p,10) then sum := sum + p
     i := i+1
     p := makePalindromeBase2(i,true)

i := 1
p := makePalindromeBase2(i,false)
while p < limit
     if isPalindrome(p,10) then sum := sum + p
     i := i+1
     p := makePalindromeBase2(i,false)
output sum
```

The presented program can handle limits as high as 1,000,000,000 in less than 20 milliseconds using a compiled language.

Copyright Project Euler, further distribution without the consent of the author(s) prohibited
Author: hk