```
-- integers: The infinite list of integers, ordered
--          as 0,1,-1,2,-2,3,-3,..

integers :: [Integer]

integers = combine [0,-1..] [1..]

--combine ns ms : Joins two non empty, same length lists together, one element
--              from each at a time.

combine :: [a] -> [a] -> [a]

combine ns ms = (head ns): (head ms): (combine (tail ns) (tail ms))

--runs xs: The number of blocks of adjacent equal items in the finite list 'xs'

runs :: Eq a => [a] -> Int

runs xs = if null xs then 0 else run (head xs) (tail xs) 1

--run n ns count: increases the 'count' by the number times elements are not
--equal to their adjacent element in list 'ns' where 'n' is the first
--element to be compared

run :: Eq a => a -> [a] -> Int -> Int

run n ns count = if null ns then count else
```

```
                 if (n == head ns) then

                 run (head ns) (tail ns) count else

                 run (head ns) (tail ns) (count + 1)


--occurrences xs: The list of tuples of distinct items

--        and their frequences in the finite list 'xs'


occurrences :: Eq a => [a] -> [(a, Int)]


occurrences xs = occurrences' (head xs) xs


--occurrences' x xs : the list of tuples of distinct items

--        and their frequences in the finite list 'xs', where 'x' is

--        the first item to be compared


occurrences' :: Eq a => a -> [a] -> [(a, Int)]


occurrences' x xs = if null xs then [] else (x, (length( filter (\n -> n == x)

                  xs))):occurrences (filter (\n -> not(n == x)) xs)


--Thanks
```