/\* Final cheatsheet

\* 目录

\* 1.i[sPrim](#isprime)e 检测参数是否为质数

\* 2.[properDiv](#pd)s 除本身以外的因子

\* 3.[primeDivisors](#pmd) 除数并且是质数

\* 4.[fibonacci](#fib)

\* 5.i[ntToLis](#intll)t 把数字转成表

\* 6[.sieve](#sie) 筛选法做出无限质数表

\* 7[.qsort](#qs) 快速排序模板

\* 8.[countApperan](#cnta)ce

\* 9.gcd

\* 10.[lcm](#lcm)

\* [11.OddthEventh](#oe) 针对list偶次位和积次位的元素做不同操作

\* 12[.factorial](#fa)

\* 13. [listToInt](#liint)

\* 14. [listToArr](#listtoarr)

\* 15. arrTOList

\* 16. goL goR

\* 17. extract node

\* 18. treetolist

\* 19. rosetolist

\*/

[1. 检测参数是否为质数](#_top)

isPrime :: Int -> Bool

isPrime n

| n < 2 = False

= isEmpty [x\\x<-[2..n-1] | n rem x == 0]

[2.除本身以外的因子](#_top)

properDivs :: Int -> [Int]

properDivs n = [div \\ div<-[1..(n-1)] | n rem div == 0]

[3. 除数并且是质数](#_top)

primeDivisors :: Int -> [Int]

primeDivisors n = [x \\ x<-[2..n] | n rem x == 0 && isPrime x]

[4. fibonacci](#_top)

从 0th 开始编号的 （ 1, 1, 2, 3, 5...）

（0th,1th,2th,3th,4th...）

fib :: Int -> Int

fib n = fibAux n 1 1

where

fibAux 0 a b = a

fibAux i a b | i > 0 = fibAux (i-1) b (a+b)

[5. 把数字转成表](#_top)

\* 没有reverse

intToList :: Int -> [Int]

intToList a

| a < 10 = [a]

= intToList (a / 10) ++ [a rem 10]

[6. sieve](#_top)

call: sieve [2..]

sieve :: [Int] -> [Int]

sieve [p:ps] = [p: sieve [i \\ i<-ps | i rem p <> 0]]

[7. qsort](#_top)

qsort :: [Int] -> [Int]

qsort [] = []

qsort [x:xs] = qsort (filter ((>)x) xs) ++ [x] ++ qsort (filter ((<)x) xs)

[8. countApperance](#_top)

countApperance :: Int [Int] -> Int

countApperance n list = length [x \\ x <- list | x == n]

9. gcd

gcd :: Int Int -> Int

gcd 0 b = b

gcd a b = gcd (b rem a) a

[10. lcm](#_top)

lcm :: Int Int -> Int

lcm x y = x \* (y/gcdxy)

where

gcdxy = gcd (min x y) (max x y)

gcd :: Int Int -> Int

gcd 0 b = b

gcd a b = gcd (b rem a) a

/\* 11. 针对list偶次位和积次位的元素做不同操作

\* 1，3，5，...进行 A(x)

\* 0，2，4，6...进行 B(x)

\* 第一个参数是表中表，第二个参数是起始的位置，建议 0 。

\*/

// OddthEventh :: [[Int]] Int -> [[Int]]

// OddthEventh x n

// | n == length x = []

// | n rem 2 == 1 = [A (x!!n) : OddthEventh x (n+1)]

// = [B (x!!n) : OddthEventh x (n+1)]

/\* [factorial](#_top)

factorial :: Int -> Int

factorial n = prod [1..n]

/\* [13. listToInt](#_top)

listToInt :: [Int] -> Int

listToInt list = sum [x\*(10^k) \\ x <- revlist & k <- [0..]]

where revlist = reverse list

////////////////////////////////////////////////////////////////////////////////////

// From here for Trees Arrays Records

/\* 14. l[istToArr](#_top)

listToArr :: [a] -> {a}

listToArr li = {x \\ x<-li}

/\* 15. arrTOList

arrTOList :: {a} -> [a]

arrTOList arr = [x\\x<-:arr]

/\* 16. goL goR

goL :: (Tree a) -> (Tree a)

goL (Node x l r) = l

goR :: (Tree a) -> (Tree a)

goR (Node x l r) = r

/\* 17. extract node

extractNode :: (Tree a) -> a

extractNode (Node x \_ \_ ) = x

/\* 18. treetolist

treeToList :: (Tree a) -> [a]

treeToList Leaf = []

treeToList (Node x l r) = treeToList l ++ x ++treeToList r

/\* 19. rosetolist

roseToList :: (ColoredRoseTree a) -> [(NodeColor, a)]

roseToList Leaf = []

roseToList (Node n color list) = [(color,n)] ++ flatten (map roseToList list)

equalDis :: (Int,Int) (Int,Int) Int -> Bool

equalDis (x,y) (a,b) dis = ((x-a)\*(x-a) + (y-b)\*(y-b) == dis\*dis)