```haskell
module Seqs(
insert,
delete,
delperms,
insperms,
isperm,
--issuf,
ispref,
--isseg,
--issubseq,
permsort,
allsufs,
sieve,
primes,
from
)
where


insert x xs = [(take n xs)++x:(drop n xs) | n<-[0..length xs]]


delete xs = [(xs!!i,(take i xs)++(drop (i+1) xs)) | i <-[0..(length xs-1)]]

delperms [x] = [[x]]
delperms xs = concat [map (y:) (delperms ys) | (y,ys) <- delete xs]

insperms [x] = [[x]]
insperms (x:xs) = concat (map (insert x) (insperms xs))

ismember x [] = False
ismember x (y:ys) = if x == y
                    then True
                    else ismember x ys

sorted pred [] = True
sorted pred [x] = True
sorted pred (x0:x1:xs) = if (pred x0 x1)
                         then sorted pred (x1:xs)
                         else False

permsort pred = head.(filter (sorted pred)).insperms


delx x [] = []
delx x (y:ys) = if x==y
                then ys
                else y:(delx x ys)

isperm [x] [y] = x == y
isperm [] ys = False
isperm (x:xs) ys = if ismember x ys
                   then isperm xs (delx x ys)
                   else False

ispref [] ys = True
ispref (x:xs) [] = False
ispref (x:xs) (y:ys) = if x==y
                       then ispref xs ys
                       else False

allprefs xs = [take i xs | i<-[0..length xs]]
allsufs xs = [drop i xs | i<-[0..length xs]]
allsegs xs = concat (map (tail.reverse.allsufs) (tail (allprefs xs)))

from m = m:(from (m+1))

takewhile pred [] = []
takewhile pred (x:xs) = if (pred x)
                        then x:(takewhile pred xs)
                        else []
primes n = if n<2
```

```
           then []
           else takewhile (< n) (sieve (from 2))


divs m n = ((mod n m) == 0)


sieve [] = []
sieve (x:xs) = x:(filter (not.(divs x)) (sieve xs))
```