```
def even(seq):
  for number in seq:
     if (number % 2 == 0):
        yield number

def stopAt(seq,n):
  for number in seq:
     if number > n: break
     yield number

class LookaheadIterator:
    def __init__(self, iterable):
        self.iterator = iter(iterable)
        self.buffer = []
    def __iter__(self):
        return self
    def __next__(self):
        if self.buffer: return self.buffer.pop()
        else: return next(self.iterator)
    def has_next(self):
        if self.buffer: return True
        try:
            self.buffer = [next(self.iterator)]
        except StopIteration: return False
        else: return True

def buffer(seq, n):
  seq = LookaheadIterator(seq)
  while True:
    res = [next(seq) for i in range(n) if seq.has_next()]
    if res == []: raise StopIteration
    else: yield res

def conditional(seq, p):
  def bufnext(seq):
    bufnext.store = [bufnext.store[1], next(seq)]
    return bufnext.store

  bufnext.store = [0, next(seq)]

  while True:
    res = bufnext(seq)
    if p(res[1]): yield res[0]
```