```
def gray_(n):
  if n == 0: return [""]
  else:
    lower = gray_(n-1)
    return ["0"+bits for bits in lower]+["1"+bits for bits in lower[::-1]]

def gray(n):
  for g in gray_(n): yield g

def memoization(f):
  def wrapper(*args):
    if not args in wrapper.cache:
      wrapper.cache[args] = f(*args)
    else:
      print("\n### cached value for {0} --> {1}".
              format(args, wrapper.cache[args]), end='\n')
    return wrapper.cache[args]
  wrapper.cache = dict()
  return wrapper

def mgray(n):
  global gray_
  gray_ = memoization(gray_)
  return gray(n)
```