1. (60) In this problem, we will develop a function `rmnulldirs()`. It will remove all empty directories within the specified directory tree.

For instance, say `rmnulldirs()` is invoked on a directory whose only contents is a subdirectory `u`, which in turn contains further subdirectories `v` and `w`, with `v` empty. Within `u` and `w`, there are nondirectory files `zu` and `zw`, respectively. The result will be that `v` will be removed, with nothing else changed.

Note: `os.rmdir(drcy)` removes the directory whose full path name is in `drcy`.

Fill in the blanks.

```python
# descends the directory tree rooted at # rootdir, removing all empty directories
import os
def checklocalempty(dummyarg, dir, flst):
    for f in flst:
        fullf = os.path.join(dir, f)
        if blank(a)(fullf):
            subdirfls = blank(b)(fullf)
        if blank(c):
            ... blank(d)

def rmnulldirs(rootdir):
    fullrootdir = blank(e)(rootdir)
    blank(f)(fullrootdir, checklocalempty, None)
```

2. (40) In the code below, we have an iterator that fetches words one at a time from a text file. (It's actually the same as in Section 6.2.3, but as an ordinary iterator, not a generator.) Here is an example:

The file `x` consists of

```
abc de f
g
h ij
klnnopq rstuv
```

We run the code:

```python
>>> from wordfetch import *
>>> g = wordfetch(open('x'))
>>> for word in g: print word
... abc
de
f
g
h
ij
klnnopq
```

Fill in the blanks:

```python
class wordfetch:
    def __init__(self, fl):
        self.fl = fl
        # words remaining in current line
    def __iter__(self):
        return self
    def next(self):
        if self.words == []:
            line = blank(a)
            # check for end-of-file
        if line == '': blank(b)
        # remove end-of-line char
        line = line[:-1]
        self.words = blank(c)
        firstword = self.words[0]
        self.words = self.words[1]
        return firstword
```