Name: _____

Directions: MAKE SURE TO COPY YOUR ANSWERS TO A SEPARATE SHEET FOR SENDING ME AN ELECTRONIC COPY LATER.

1. (25) Consider line 27, p.70. Which of the statements below is/are true?

(i) That line is normally part of a loop.

(ii) If the length of the test string in line 18, p.71 is less than 1000000, a loop is not ncessary.

(iii) Neither (i) nor (ii) is necessarily true.

2. (75) Consider the **textfile** class (original version), p.24. We add an instance method to that class, **getnlines1tf()**. (This is what is known as a *getter* in the OOP world. I generally think getters and setters are silly, but it will be helpful here.)

In the following, the argument **tfl** ("**textfile** list") is a list of **textfile** objects, and **k** is a positive integer. Fill in the blanks:

```
class textfile:
   # ... as before
   def getnlines1tf(self):
      return self.nlines

# return list, element i of which is
# the number of lines in tfl[i]
def getnlines(tfl):
   return map(  blank (a)  )

# return the total number of lines in all
# the files in tfl
def totlines(tfl):
   tmp = getnlines(tfl)
   return  blank (b)

# return sublist of tfl, element i of which is
# the i-th element of tfl that satisfies the
# condition (number of lines > k)
def bigfiles(tfl,k):
   return  blank (c)

# sort tfl in-place, according to the number
# of lines in each file
def tflsort(tfl):
   blank (d)

def test():
   a = textfile('x')
   b = textfile('y')
   c = textfile('z')
   tflist = [a,b,c]
   print getnlines(tflist)
   print totlines(tflist)
   print tflist
   print bigfiles(tflist,3)
   tflsort(tflist)
   print tflist
```

Here in the input to the test case:

```
% cat x
a
bc
def
% cat y
1234
456
78
9
% cat z
a1b2
c3
```

Here is the output:

```
% python tfclass.py
[3, 4, 2]
9
[<__main__.textfile instance at 0x1054b0290>,
   <__main__.textfile instance at 0x1054b02d8>,
   <__main__.textfile instance at 0x1054b0320>]
[<__main__.textfile instance at 0x1054b02d8>]
[<__main__.textfile instance at 0x1054b0320>,
    <__main__.textfile instance at 0x1054b0290>,
    <__main__.textfile instance at 0x1054b02d8>]
```

Solutions:

1. (i)

2.

```
class textfile:
   ntfiles = 0  # count of number of textfile objects
   def __init__(self,fname):
      textfile.ntfiles += 1
      self.name = fname  # name
      self.fh = open(fname)  # handle for the file
      self.lines = self.fh.readlines()
      self.nlines = len(self.lines)  # number of lines
      self.nwords = 0  # number of words
      self.wordcount()
   def wordcount(self):
      "finds the number of words in the file"
      for l in self.lines:
         w = l.split()
         self.nwords += len(w)
   def grep(self,target):
      "prints out all lines containing target"
      for l in self.lines:
         if l.find(target) >= 0:
            print l
   def getnlines1tf(self):
      return self.nlines

def getnlines(tfl):
   return map(lambda onetf: onetf.getnlines1tf(),tfl)

def totlines(tfl):
   tmp = getnlines(tfl)
   return reduce(lambda x,y: x+y,tmp)

def bigfiles(tfl,k):
   return filter(lambda x: x.nlines > k,tfl)

def tflsort(tfl):
   tfl.sort(lambda x,y: x.nlines - y.nlines)

def test():
   a = textfile('x')
   b = textfile('y')
   c = textfile('z')
   tflist = [a,b,c]
   print getnlines(tflist)
   print totlines(tflist)
   print tflist
   print bigfiles(tflist,3)
   tflsort(tflist)
   print tflist
```