Question 1. (10 points) For the following program, predict the output.

```python
cheer = 'GO Panthers!!!
ryhme = [1, 2, 'buckle', 'your', 'shoe']
print 'cheer:', cheer[4]
print 'ryhme:', rhyme[4]
ryhme.append('cat'.upper())
print 'more rhyme:', rhyme[-3:]
```

```python
cheer:
ryhme: shoe
more rhyme: your
```

2. (20 points) Below is a summary of some file-system functions in Python.

<table>
<thead>
<tr>
<th>General syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>os.getcwd()</td>
<td>Returns the complete path of the current working directory</td>
</tr>
<tr>
<td>os.chdir(path)</td>
<td>Changes the current working directory to path</td>
</tr>
<tr>
<td>os.listdir(path)</td>
<td>Returns a list of the names in directory named path</td>
</tr>
<tr>
<td>os.rename(old, new)</td>
<td>Renames the file or directory named old to new</td>
</tr>
<tr>
<td>os.path.exists(path)</td>
<td>Returns True if path exists and False otherwise</td>
</tr>
<tr>
<td>os.path.isdir(path)</td>
<td>Returns True if path is a directory and False otherwise</td>
</tr>
<tr>
<td>os.path.isfile(path)</td>
<td>Returns True if path is a file and False otherwise</td>
</tr>
<tr>
<td>os.path.getsize(path)</td>
<td>Returns the size in bytes of the object named path</td>
</tr>
</tbody>
</table>

Write statements to display only the files (but not the directories) in the current working directory.
(For extra credit, order the files by size from the largest number of bytes to the smallest.)

```python
dir_list = os.listdir('.')
for item in dir_list:
    if os.path.isfile(item):
        print 'file:', item
```
Question 3. (15 points) Predict the output of the Python code segment.

```python
def bar(a, b=8, *args, **kwargs):
    print 'a=',a,'b=',b,'args=',args,'kwargs=',kwargs
    a += a
    b=2

s=1
t=['x','y']
myTuple = ('pi', 3.14)
bar(myTuple, s=11, t=12)
bar(s=11, t=12, a=4, b=6)
bar(t, s, t, d=4)
print 'myTuple =', myTuple, 't =',t, 's =', s
```

<table>
<thead>
<tr>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a = ('\pi', 3.14) ) ( b = 8 ) ( \text{args} = () ) ( \text{kwargs} = { 's': 11, 't': 12 } )</td>
</tr>
<tr>
<td>( q = 4 ) ( b = 6 ) ( \text{args} = () ) ( \text{kwargs} = { 's': 11, 't': 12 } )</td>
</tr>
<tr>
<td>( q = ['x', 'y'] ) ( b = 1 ) ( \text{args} = ( ['x', 'y']) ) ( \text{kwargs} = { 'd': 4 } )</td>
</tr>
<tr>
<td>( \text{myTuple} = ('\pi', 3.14) ) ( \text{x} = ['x', 'y', 'x', 'y'] ) ( s = 1 )</td>
</tr>
</tbody>
</table>

Question 4. (10 points) Predict the output of the following higher-order functions. Assume `double` is defined as:

```python
def calculate(a, b):
    return a + 2 * b
```

a) `print reduce(calculate, range(1,5))`

b) `print map(lambda s: s.lower(), ['YES', 'Yes', '2'])`

c) `print filter(lambda x: x%10 == 0, range(50))`

\[ [0, 10, 20, 30, 40] \]

d) Rewrite the "reduce(calculate, range(1,5))" from part (a) using a lambda expression instead of the function calculate.

```
print reduce(lambda a, b: a+2*b, range(1,5))
```
Question 5. Professor Plum has a menu-driven, grade-book program that maintains a text file gradeBook.txt of student records. Each student record is on a single line with values separated by commas (','). The order of information on a line is: a unique student ID number (a string of 6 digits), first name, last name, and some number of scores. The first line in the file contains comma separated titles for each field of a student record. For example, the start of the gradeBook.txt file might look like:

```
ID#, First, Last, HW1, Lab1, HW2, Test1, Lab2, HW3, Lab3
123456, John, Doe, 12, 5, 10, 84, 5, 10, 12
345678, Sue, Smith, 11, 4, 9, 90, 0, 12, 10
```

When the program starts, the file gradeBook.txt is read into two lists:
- titlesList - a list of string titles for each field of a student record
- studentsList - a list of lists with each item in the list containing the information about a student as a list

For the above file, the titlesList would look like:
```
['ID#', 'First', 'Last', 'HW1', 'Lab1', 'HW2', 'Test1', 'Lab2', 'HW3', 'Lab3']
```

For the above file, the studentsList would look like:
```
[['123456', 'John', 'Doe', '12', '5', '10', '84', '5', '10', '12'],
 ['345678', 'Sue', 'Smith', '11', '4', '9', '90', '3', '12', '10']], ...
```

The main function for this grade-book program is:
```python
def main():
    titlesList, studentsList = readGradeBook('gradeBook.txt')
    mainMenu(titlesList, studentsList)
    writeGradeBook('gradeBook.txt', titlesList, studentsList)
```

a) (15 points) Complete the following readGradeBook function that takes in a file name as a string and returns the titlesList and studentsList.
```python
def readGradeBook(fileName):
    gradeFile = open(fileName, 'r')
    studentsList = []
    for line in gradeFile:
        record = line.strip().split(',')
        studentsList.append(record)
    titlesList = studentsList[0]
    studentsList = studentsList[1:]
    return titlesList, studentsList
b) (15 points) Assuming `titlesList` and `studentsList` has been read in from the file successfully. One of the main menu operations is to generate a table consisting of two columns: student name and their corresponding total score. To aid in generating this table, you are to write a function `sumStudentScores` that takes a single student record list as its parameter and returns the total of the student's scores.

```python
def sumStudentScores(studentInfo):
    Sum = 0.0
    for index in xrange(3, len(studentInfo)):
        Sum += float(studentInfo[index])
    return Sum
```

c) (15 points) Write the function `generateGradeReport` which uses the above function (e.g., `sumStudentScores`) to generate a table consisting of two columns: student name and their corresponding total score. The function `generateGradeReport` takes the `studentsList` as a parameter and writes the table to the file named `gradeReport.txt`. (NOTE: You can answer this question even if you did not complete part (b))

```python
def generateGradeReport(studentsList):
    reportFile = open('gradeReport.txt', 'w')
    reportFile.write('%35s%5s
' % ('Student Name', 'Total'))
    reportFile.write('-1x43 + \n')
    for student in studentsList:
        total = sumStudentScores(student)
        reportFile.write('%35s%5.1f
' % (student[2] + ' ' + student[1], total))
    reportFile.close()
```