**PROBLEM 1 :** *(What are the types and values? (24 points))*

Consider the following variables and their values for the table below.

```python
words = ['data', 'file', 'for', 'list', 'append', 'string']
phrase = '3 ants 5 cats 2 pigs'
```

List in the table the type of variable and its value after being assigned the expression.

<table>
<thead>
<tr>
<th>variable = expression</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a = 'giraffe'[5]</td>
<td>string</td>
<td>'f'</td>
</tr>
<tr>
<td>b = len(words[3])</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c = phrase[3] + phrase[-2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d = words[0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e = phrase.find('t')</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f = phrase.split(&quot;a&quot;)[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g = words[4][3:]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>h = &quot;dog&quot;.join([&quot;a&quot;,&quot;cat&quot;])</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i = &quot;-&quot;.join(list(&quot;bus&quot;))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>j = len(phrase.split()) &gt; len(words)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>k = words[2:4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 4.2 + 7/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n = 8 % 5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PART A: Discount (6 pts)

At the Super Discount Store, an item is discounted based on how many days it has been on sale, by multiplying the number of days times the percent for one day. If an item has been on sale 3 days, and the percentage discount is 10 percent per day, then 30 percent discount is taken off the regular price. If the percent off is more than 100 percent, the item has no cost. It is free, so return a cost of 0.

Write the function `calculate` that has three parameters: `price` is a float representing the regular cost of the item when there is no sale, `days` is an integer representing the number of days the item has been on sale, and `percent` is an integer representing the percentage off for one day. This function returns the sale price of the item as a decimal number.

<table>
<thead>
<tr>
<th>call</th>
<th>returns</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculate(100.00, 7, 10)</td>
<td>30.0</td>
<td>70 percent off the regular price</td>
</tr>
<tr>
<td>calculate(10.00, 3, 10)</td>
<td>7.0</td>
<td>30 percent off the regular price</td>
</tr>
<tr>
<td>calculate(52.00, 12, 10)</td>
<td>0.0</td>
<td>120 percent off the regular price</td>
</tr>
</tbody>
</table>

def calculate(price, days, percent):

PART B: How far (8 pts)

A football field is 100 yards long but the yards are marked from 0 to 50 yards, with 50 being the center and then counting back down from 50 to 0. We will call one side of the field the "A" side of the field and the other side the "B" side of the field. One football play goes from one location on the field to another location on the field. Your task is to compute the yardage for the one play. For example, here is the field showing every fifth yard:

```
"A" side     | "B" side
0 5 10 15 20 25 30 35 40 45 50 45 40 35 30 25 20 15 10 5 0
```

Write the function `yardage` that has four parameters: `stfield` is a string indicating which side of the field the ball starts (either "A" or "B"), `styard` is an integer from 0 to 50 indicating the yard line the ball starts on, `efield` is a string indicating the side of the field the ball ends at, and `eyard` is an integer indicating the yard line the ball is on at the end of the play. This function returns total yardage for the play.

def yardage(stfield, styard, efield, eyard):
call | returns | comment
---|---|---
yardage("A", 24, "A", 48) | 24 | from A24 to A48, 24 on A side
yardage("A", 29, "B", 35) | 36 | from A29 to B35, 21 on A side and 15 on B side
yardage("B", 30, "A", 45) | 25 | from B30 to A45, 20 on B side and 5 on A side

PROBLEM 3:  (It’s a mystery and debugging (12 points))

PART A: Mystery (6 pts)
Consider the following mystery function that has three parameters, where `words` is a list of strings, and `let1` and `let2` are both strings. The lines have been numbered.

```
1 def mystery(words, let1, let2):
2     ans = []
3     for w in words:
4         pos = w.find(let1)
5         pos2 = w.find(let2)
6         if pos2 > pos:
7             ans.append(w)
8         else:
9             ans = [w] + ans
10 # print ans
11 return ans
```

Q1. Consider the call to mystery.

```
wordlist = ["fish", "tigers", "street", "eels", "strike"]
result = mystery(wordlist, "e", "s")
```

For this sample call, what is the value assigned to result?

Q2. Suppose the print statement on line 10 is uncommented. For the sample call in Q1, what is the value printed the first time this print statement is executed?

Q3. Suppose the print statement on line 10 is uncommented. For the sample call in Q1, what is the value printed the second time this print statement is executed?

Q4. Explain in words what this function does, that is, what does it calculate for any given inputs?

PART B: Debugging (6 points)
Consider the attempt to implement the function `onlyOneVowel` that has one string parameter `word`. This function is supposed to return the location of the vowel in word if there is
only one vowel. If there is more than one vowel in word (or no vowels), the function should return -1. **This function does not work correctly!**

Note that `onlyOneVowel` calls the function `isVowel` that has one string parameter `ch`. The function `isVowel(ch)` returns True if `ch` is a vowel and False if it is not. You can assume that `isVowel` works as intended.

```python
def onlyOneVowel(word):
    count = 0
    for ch in word:
        if isVowel(ch):
            count += 1
    if count == 1:
        return word.find(ch)
    else:
        return -1
```

Here are two calls to `onlyOneVowel`, one with a correct answer and one with a wrong answer.

<table>
<thead>
<tr>
<th>call</th>
<th>returns</th>
<th>correct answer</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>onlyOneVowel(&quot;boat&quot;)</td>
<td>-1</td>
<td>-1</td>
<td>more than one vowel</td>
</tr>
<tr>
<td>onlyOneVowel(&quot;string&quot;)</td>
<td>5</td>
<td>3</td>
<td>one vowel, wrong position returned</td>
</tr>
</tbody>
</table>

Q1. Explain why the second call above is returning the wrong answer, 5.

Q2. Give an example of another word that returns the wrong answer

Q3. Give an example of a word of length 2 or more that has only one vowel, and that returns the correct answer.

Q4. Explain how to correct the code above so it always returns the intended answer. (you can mark your answer on the code above or just explain here).

**PROBLEM 4 : (Transformation (8 points))**

Write the function `reformat` which has two parameters, `name` is a string of words separated by blanks that represent a person’s name, and `num` is an integer. The parameter `num` is the number of words in the last name. You can assume there are always at least two words in name, the first name is one word, and `num` is greater than 0. This function returns name in a different format, with all the last names first (separated by blank), followed by a comma and a blank, followed by the first name, followed by a blank and followed by a string of the first letter of all the middle names. If there are no middle names, the name ends with the first name.

Consider these examples.

```python
def reformat (name, num):
```
PROBLEM 5:  \textbf{(Snacks before dinner (28 points))}

Consider information about snacks that is stored in a file in the following format. Each line represents information about one snack. For each line there are four pieces of information and several separators (in this order): the name of the snack (one or more words), a blank, a left parenthesis, the weight of the snack, a right parenthesis, the number of calories as an integer, a space, and the type of snack (one or more words).

Shown below is a sample file. In the first line the snack is ”Twix”, it weighs 1.02 ounces, it has 144 calories and it is of type ”chocolate”.

\begin{verbatim}
Twix (1.02)144 chocolate
apple (6.4)95 fruit
avocado (5.29)240 fruit
stick pretzels (1.5)160 chips
orange (6.34)85 fruit
honey mustard pretzels (2.25)320 chips
m&ms (1.69)230 chocolate
peach (6.17)85 fruit
chex mix (1.72)210 chips
m&ms peanut (1.74)250 chocolate
mini pretzels (1.5)160 chips
KitKat (1.5)210 chocolate
Jack Links Beef Jerky (2.5)200 salty protein
pear (5.22)86 fruit
Skinny Pop popcorn (0.65)100 chips
hershey with almonds (1.45)210 chocolate
swedish fish (1.41)140 candy
banana (4.16)105 fruit
CheezIt (1.5)210 chips
raisins (5.82)493 fruit
Reeses peanut butter cups (1.50)210 chocolate
\end{verbatim}

A. (6 pts)
Write the function \textit{getParts} that has one parameter, a string in the format of one line from the previous page representing one snack, and \textbf{returns a list of four items} of information about that snack: the name of the snack as a string, the number of ounces as a float, the number of calories as an integer and the type of snack as a string.

\begin{verbatim}
def getParts(line):
\end{verbatim}
B. (4 pts)
Consider the function named `fileToList` that has one string parameter `filename` that represents a datafile in the format on page 8 and returns a list of lists in the format shown below, where each list in the big list has four pieces of information representing one line from the data file (and thus one snack): the name of the snack as a string, the number of ounces as a float, the number of calories as an integer and the type of snack as a string.

The line, `datalist = fileToList("snackdata.txt")` where `snackdata.txt` is the file on page 8 results in the list of lists (not all shown):

```python
datalist = [
    ['Twix', 1.02, 144, 'chocolate'],
    ['apple', 6.4, 95, 'fruit'],
    ['avocado', 5.29, 240, 'fruit'],
    ...
    ['Reeses peanut butter cups', 1.5, 210, 'chocolate']
]
```

The function is started below and has some missing lines indicated by MISSING LINES. The function should read in all the lines from the file and return the lists of lists in the format described above. **You must call the function getParts from part A in this function.** Add in the missing lines.

```python
def fileToList(filename):
    answer = []
    f = open(filename)
    #MISSING LINES
    return answer
```

C. (8 points) Write the function `snacksOfType` that has three parameters: `snklist` is the list of lists in the format described in Part B, where each list has four items, `type` is a string representing the type of snack, and `amount` is an integer.

This function returns a list of snack items from `snklist` that are the same type as the parameter `type` and have the same or fewer number of calories then `amount`.

Assume `datalist` is the example list of lists created in Part B from the datafile on page 8. Then the call `snacksOfType(datalist, "chips", 160)` returns the list `['stick pretzels', 'mini pretzels', 'Skinny Pop popcorn']`
D. (10 points) Write the function `mostCalories` which has two parameters: one named `snklist` where `snklist` is a list of lists in the format described in Part B (where each list is four items representing one snack), and a string parameter named `type`. This function returns the name of the snack of type `type` that has the most calories. If there is a tie, then return any one of those that tied.

Consider the example below. Assume `datalist` is the example list of lists formed in Part B from the data file on page 8.

<table>
<thead>
<tr>
<th>call</th>
<th>returns</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mostCalories(datalist, &quot;chocolate&quot;)</code></td>
<td>&quot;m&amp;ms peanut&quot;</td>
</tr>
</tbody>
</table>

def mostCalories(snklist, type):