

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

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

```
words = ['cat', 'fish', 'turtle', 'pig', 'dog']
phrase = 'from 5 youngling to 5 jedi knight'
```

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

variable = expression	Type	Value
a = "CompSci"	string	"yes"
b = phrase[-2]+phrase[2]		
c = words[1] + words[-1]		
d = 1.0 + 6/4		
e = 11 % 3		
f = 9.0/2 > 4.0		
g = words[2][:2]		
h = words[0].upper()		
i = phrase.split()[-1]		
j = phrase[5:9] + "giraffe"[:2]		
k = phrase.find("gi")		
m = "\$".join(['co','a','5'])		
n = len(words) + len(phrase.split())		

PROBLEM 2 : (*How Many? - Simple Function (8 points)*)

The restaurant *Durham Durham* has lots of discounts on meals.

1. On Mondays, the lower cost item is 50% off. On Tuesdays, the lower cost item is 10% off.
2. In April and May you get \$5 off your meal after other discounts, but note the total should not be negative.

Write the function `dinnerPrice` that has four parameters: **item1** and **item2** are floats representing the cost of two food items, **day** is a string representing the current day of the week, and **month** is a string representing the current month. This function returns the total cost of dinner after applying discounts explained above.

call	returns	comment
<code>dinnerPrice(8.5, 4.0, "Wednesday", "May")</code>	7.5	\$5 off for May, $8.5 + 4.0 - 5 = 7.5$
<code>dinnerPrice(8.5, 4.0, "Monday", "January")</code>	10.5	50% off 4.0, $8.5 + 2.0 = 10.5$
<code>dinnerPrice(2.5, 2.0, "Monday", "April")</code>	0.0	50% off 2.0, \$5 off for April
<code>dinnerPrice(5.0, 10.0, "Tuesday", "May")</code>	9.5	10% off 5.0, \$5 off for May

```
def dinnerPrice(item1, item2, day, month):
```

PROBLEM 3 : *(It's a mystery and debugging (14 points))*

PART A: Mystery (6 pts)

Consider the following mystery function that has three parameters, where `wordlist` is a list of strings, `value` is an integer, and `ch` is a string. The lines have been numbered.

```
1 def mystery(wordlist, value, ch):
2     ans = []
3     for w in wordlist:
4         if len(w) > value:
5             ans.append(w)
6         elif w[0] == ch:
7             ans.append(w)
8         #print ans          # line is commented out
9     return ans
```

Q1. Consider the call to `mystery`.

```
wordlist = ["apple", "fig", "lime", "honeydew", "avocado", "lemon"]
result = mystery(wordlist, 6, "a")
```

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

Q2. Suppose the print statement on line 8 is uncommented. For the sample call in Q1, what is the value printed the first time this print statement is executed?

Q3. Explain in words what this function does, that is, what does it calculate for any given inputs?

PART B: Debugging (8 points)

Consider the attempt to implement the `find` function with two string parameters, *phrase* and *small*. The function `find` is suppose to return the index location of the first occurrence of *small* in *phrase*, or -1 if *small* does not occur in *phrase*. **This function does not work correctly!**

```
1 def find(phrase,small):
2     pos = -1
3     for i in range(len(phrase)):
4         ch = phrase[i]
5         if ch == small:
6             pos = i
7     return pos
```

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

```
phrase = "the circle circus is closing"
```

call	returns	correct answer
<code>find(phrase, "c")</code>	21	4
<code>find(phrase, "h")</code>	1	1

Q1. Explain why the first call above is returning the wrong answer, 21.

Q2. Explain how to correct the code above by changing one line so it always returns the intended answer when `small` is a single character. Give the line number and the new code.

Q3. Consider the call `find(phrase, "cir")` with the original code above. It should return 4, but does not work correctly. What does it return?

Q4. Given your code change in Q2, explain how to correct the code above with one additional line change so it always returns the intended answer when `small` is of size greater than 1. Give the line number and the new code.

PROBLEM 4 : (*Transformations (16 points)*)

PART A (8 pts): Write the function `allIn` which has two string parameters `word` and `letters`. This function returns `True` if all the characters in `letters` appear in `word`. You can assume the characters in `letters` are all unique. Consider these examples.

call	returns	comment
<code>allIn("creatures", "eur")</code>	<code>True</code>	e, u and r are in creatures
<code>allIn("creatures", "atcd")</code>	<code>False</code>	d is not in creatures; a, t and c are in
<code>allIn("computer", "eurmt")</code>	<code>True</code>	e, u, r, m and t are all in computer

```
def allIn(word, letters):
```

PART B (8 pts): Write the function `message` which has two string parameters `phrase` and `code` and returns a secret message. One letter is selected from each word in `phrase` to form the secret message. For each word, if all the letters in `code` are in the word, select the first letter of that word. If not, select the last letter of that word.

For example, if `phrase` is "bloomed fire lemon leopard sole" and `code` is "ole", then the secret message is "bells". The b comes from the first letter of "bloomed" since o, l and e are all in "bloomed". The e comes from the last letter in "fire" since o and l are not in "fire". The first l comes from the first letter in "lemon", since o, l and e are in "lemon", etc.

For full credit, you must call the function `allIn` you wrote in Part A. Assume it works correctly.

```
def message(phrase, code):
```

PROBLEM 5 : (*Where to eat dinner? (24 points)*)

Consider information about restaurants that is stored in a file in the following format. Each line represents information about one customer who ate at a particular restaurant. For each line there are four pieces of information and three separators (in this order): the name of the customer, a blank, the customer's rating of the restaurant, a colon, the name of the restaurant, a \$, and the price of the meal for that person. Note that ratings are an integer and cost of meals are a decimal number.

Shown below is a sample file. In the first line the customer is "Bradley Atkinson", he has rated the restaurant "Sushi Love" a 16, and the cost of his meal there was 18.30.

```
Bradley Atkinson 16:Sushi Love$18.30
Gini Carlson 11:Another Broken Egg Cafe$8.64
Ming Lao Zhang 15:Parizade$27.87
Gini Carlson 18:Nana Tacos$9.98
Ming Lao Zhang 18:The Little Dipper$35.76
Emily Sue Lynn Moon 17:Pompieri Pizza$28.53
Bala Yavatkar 14:Qshack$14.56
Bala Yavatkar 18:Nana Tacos$15.72
Ming Lao Zhang 12:The Little Dipper$34.74
Gini Carlson 19:Nana Tacos$11.54
Bradley Atkinson 15:Sushi Love$22.75
Emily Sue Lynn Moon 13:The Little Dipper$36.43
Ming Lao Zhang 12:Parizade$24.18
Bala Yavatkar 12:The Little Dipper$41.93
Emily Sue Lynn Moon 16:Nana Tacos$13.81
Gini Carlson 14:Pompieri Pizza$18.65
Bala Yavatkar 11:Qshack$12.76
Emily Sue Lynn Moon 18:Nana Tacos$7.76
```

A function has been written named `fileToList` that reads in a datafile in the format above 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: the customer as a string, the rating as an integer, the restaurant as a string, and the amount of the meal as a decimal number. That list of lists created for the file above is partly shown below.

The line, `datalist = fileToList("ratings.txt")` where `ratings.txt` is the file above results in

```
datalist = [ ['Bradley Atkinson', 16, 'Sushi Love', 18.3],
             ['Gini Carlson', 11, 'Another Broken Egg Cafe', 8.64],
             ['Ming Lao Zhang', 15, 'Parizade', 27.87],
             ...
             ['Emily Sue Lynn Moon', 18, 'Nana Tacos', 7.76] ]
```

A. (4 pts) Consider the function `fileToList` which has one parameter, `filename` that is the name of a file that is in the format on the previous page. This function reads in the file and returns a list of lists in the format also on the previous page, where each list inside the larger list is **four items** representing one customer's meal in one restaurant. In particular those four items are the customer as a string, the rating as an integer, the restaurant as a string, and the amount of the meal as a decimal number.

For example, the string "Bala Yavatkar 12:The Little Dipper\$41.93" is replaced as the list `['Bala Yavatkar', 12, 'The Little Dipper', 41.93]`.

The function is below and has one or more missing lines indicated by MISSING LINE(S).

Add in the missing line(s).

```
def fileToList(filename):  
    answer = []  
    f = open(filename)  
    for line in f:  
        line = line.strip()  
        #MISSING LINE(S)
```

```
    return answer
```

B. (10 points) Write the function `dinersForRestaurant` that has two parameters named `datalist` and `restname`, where `datalist` is a list of lists in the format described on the first page of this problem (each list has four items), and `restname` is a string representing the name of a restaurant. This function returns the **unique** list of names of people who ate at the restaurant `restname`.

Consider the two examples below. Assume `datalist` is the example list of lists on the bottom of the first page of this problem from the datafile given. Note that **Gini Carlson** ate at **Nana Tacos** twice, but just appears in the list once.

call	returns
<code>dinersForRestaurant(datalist,"Nana Tacos")</code>	<code>['Gini Carlson', 'Bala Yavatkar', 'Emily Sue Lynn Moon']</code>
<code>dinersForRestaurant(datalist,"Pompieri Pizza")</code>	<code>['Emily Sue Lynn Moon', 'Gini Carlson']</code>

```
def dinersForRestaurant(datalist, restname):
```

C. (10 points) Write the function `restWithHighestRating` which has one parameter named `datalist` where `datalist` is a list of lists in the format described on the first page of this problem (where each list is four items representing a person eating at a restaurant). This function returns the **name of the restaurant that has the highest rating**. 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 on the first page of this problem from the datafile given. Note that in that file, the restaurant with the highest rating is **Nana Tacos**, with a rating of 19.

call	returns
<code>restWithHighestRating(datalist)</code>	<code>"Nana Tacos"</code>

```
def restWithHighestRating(datalist):
```