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

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

```
words = ['red', 'green', 'tree', 'play']
str = 'Oh! no! not now!'
```

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

variable = expression	**Type**	**Value**
`a = "yes"`	string	`"yes"`
`b = str[2]`		
`c = words[-1]`		
`d = 13/4`		
`e = 5 + 2.0 * 4`		
`f = len(words)`		
`g = words[2][1]`		
`h = words[1].upper()`		
`i = str.split()[1]`		
`j = str[5:6] + str[:2]`		
`k = str.find("not") > str.find("now")`		
`m = "y".join(['mo','bo'])`		
`n = 24 % 7`		

PROBLEM 2 : (***How Many? - Simple Functions (14 points)***)

A. (6 pts) Consider the problem of calculating how many of a particular item one could purchase.

Write the function `howMany` that has three parameters: `amount` is a decimal number representing the total amount of money one has, `discount` is an integer representing the discount one gets, and `cost` is a decimal number representing the cost per item. This function returns the number of items one could purchase.

In the first example below, one has $7.42, and items cost $1.00 each but there is a discount of 30%, meaning items actually cost $0.70 each. One could purchase ten of the items costing $7.00. There is not enough money to purchase any more.

call	**returns**	**comment**
howMany(7.42, 30, 1.00)	10	30% discount means items cost .70 each
howMany(5.4, 10, 2.0)	3	10% discount means items cost 1.80
howMany(10.0, 50, 5.0)	4	50% discount means items cost 2.50

```
def howMany(amount, discount, cost):
```

B. (8 pts) You want to purchase a car. You have determined the following formula is used to sell cars at a particular dealership.

1. You get a discount of 10 percent if you are purchasing a car on the last day of a week (day 7, day 14, day 21 or day 28) for this month.
2. After taking the discount in 1), you get an additional $1000 off if you purchase a car the last 5 days of the month

Write the function `carCost` that has three parameters: **price** is a float that is the cost of the car without discounts, **day** is the integer numbered day of the week for the current month, and **numDays** is the integer number of days in the current month. This function returns the cost of the car after applying discounts explained above.

Examples of calls are shown.

```
def carCost(price, day, numDays):
```

call	returns	comment
carCost(30000.0, 14, 30)	27000.0	10% discount for day 14
carCost(30000.0, 27, 28)	29000.0	$1000 off bought in last 5 days of month
carCost(30000.0, 28, 31)	26000.0	Both 10% and $1000 discounts

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

PART A: Mystery (6 pts)

Consider the following mystery function that has two parameters, where `words` is a list of Strings, and `item` is a string. The lines have been numbered.

```
1 def mystery(words, item):
2     answer = []
3     count = 0
4     for w in words:
5         newword = w
6         if count % 2 == 0:
7             newword = item
8         answer.append(newword)
9         count += 1
10    return answer
```

Consider the call to mystery.

```
words = ['apple', 'fig', 'kiwi', 'lemon', 'lime']
result = mystery(words,'peach')
```

Q1. For the sample call to mystery above, what is the value assigned to result?

Q2. For that same call, what is the first value assigned to w in the `for` loop on line 4.

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

PART B: Debugging (6 points)

Consider the following `IsWordInPhrases` function that has two parameters, where `listPhrases` is a list of Strings, and `word` is a String. The lines have been numbered.

This function is suppose to return True if word is a word in any of the phrases, and return False otherwise, but does not work correctly.

```
1 def IsWordInPhrases(listPhrases, word):
2     for phrase in listPhrases:
3         if word in phrase:
4             return True
5     return False
```

```
listPhrases = ["cat and dog", "one green tree", "last Friday"]
```

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

call	returns	correct answer
IsWordInPhrases(listPhrases, "day")	True	False
IsWordInPhrases(listPhrases, "and")	True	True

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

Q2. Explain how to correct the code above so it always returns the intended answer.

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

PART A (8 pts): Write the function `secondDash` which has one string parameter `word`. This function returns the location of the second occurrence of the dash ('-') in the word. If the dash does not occur a second time, return -1. For example, in the first example below, the first dash occurs at position 2 and the second dash occurs in position 5.

call	returns
secondDash("hi-to-all")	5
secondDash("hello-all")	-1
secondDash("two-three-four-five")	9

```
def secondDash(word):
```

PART B (8 pts): Write the function `transformWord` which has one string parameter `word`. This function returns the word modified using the following rules: 1) if the word starts with a vowel, replace the first two dashes with "*"s 2) if the word does not start with a vowel, duplicate every character in word but the last character.

For full credit, you must call the provided function `isVowel` and the function secondDash you wrote in Part A.

```
def isVowel(let):
    return let in "aeiouAEIOU"
```

call	returns	comment
transformWord("hello")	"hheellllo"	duplicate all but "o"
transformWord("ap-ple")	"ap*ple"	vowel, one dash, replace it
transformWord("hi-to-all")	"hhii–ttoo–aalll"	duplicate all but last "l"
transformWord("one-two-three-four")	"one*two*three-four-five"	replace first two dashes

```
def transformWord(word):
```

PROBLEM 5 : (***It Goes to the Highest Bidder? (24 points)***)

Consider information about an auction that is stored in a file in the following format. Each line represents the bid on an object by a bidder, and there are three pieces of information about that bid. For each line, those three pieces of information are (in this order) the number of the bidder (exactly four digits), the name of the item bid on, followed by the amount bid which has a "$" followed by a decimal number. Shown is a sample file. In the first line the number of the bidder is `2237`, the object bid on is `candlestick`, and the amount bid is `20`.

```
2237candlestick$20
7451candlestick$25
8426candlestick$40
2237candlestick$45
8426candlestick$60
2237candlestick$75
4764antique table$100
2237antique table$120
6734antique table$150
8426antique table$200
6734antique table$230
2237bar stool$20
7341bar stool$30
4764bar stool$35
2237bar stool$45
6734bar stool$50
7341bar stool$75
2237bar stool$90
```

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 three strings representing the three pieces of information about one auction bid from the file above: the bidder as a string, the item as a string, and the amount bid on as a string without the $. That list of lists created for the file above is partly shown below.

The line, `datalist = fileToList("auctionData.txt")`

where `auctionData.txt` is the file above results in

```
datalist = [ ['2237', 'candlestick', '20'],
['7451', 'candlestick', '25'],
...
['2237', 'bar stool', '90'] ]
```

You will complete the method `fileToList` on the next page.

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 **three strings** representing one auction bid in the format: the four digit bidder, the item, and the amount bid on as a string without the $. For example, the string `"2237candlestick$20"` is replaced as the list `['2237', 'candlestick', '20']`.

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 `amountsBidded` that has two parameters named `data` and `bidder`, where `data` is a list of lists in the format described on the first page of this problem (each list is three strings representing one auction bid), and `bidder` is a string representing the id of a particular bidder. This function returns the **unique** list of strings of amounts the bidder has bid.

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 bidder "2237" bid the amounts 20 and 45 more than once for different items, but they each appear just once in the list.

call	**returns**
amountsBidded(datalist, "2237")	['20', '45', '75', '120', '90']
amountsBidded(datalist, "6734")	['150', '230', '50']

```
def amountsBidded(data, bidder):
```

C. (10 points) Write the function `objectWithHighestBid` which has one parameter named `data` where `data` is a list of lists in the format described on the first page of this problem (where each list is three strings representing one auction bid). This function returns the object that had the highest bid of the auction. If there is a tie, return any one of the objects with the highest bid.

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 'antique table' had the highest overall bid which was $230.00.

call	**returns**
objectWithHighestBid(datalist)	'antique table'

```
def objectWithHighestBid(data):
```