CompSci 101 **Exam 1 Sec01** **Spring 2015**

(Note 001 meets in White Lecture Hall, 002 meets in LSRC B101)

PROBLEM 1 : (*What is the output? (20 points)*)

A. (10 pts) What is the output of the following code segment? Write the output to the right. Note that there is only output for the print statements.

```
                                            Output:
bo = 7
mo = 2
x = 1.5
print type(x)
print bo - 3 * mo
print bo <= mo
print pow(bo, 2) / 10
print bo * 2 % 6
```

B. (10 pts) What is the output of the following code segment? Write the output to the right. Note that there is only output for the print statements.

```
                                                      Output:
city = "Washington"
print city[2]
pos = city.find("n")
print city[pos-3:pos]
print city[-3:]

cities = ["Durham","Atlanta",city,"Chicago","Cary"]
print cities[-1]
print cities[1:3]
```

PROBLEM 2 : (*Money Conversion - Simple Functions (14 points)*)

A. (6 pts) Consider the problem of exchanging currency, in particular, Rupees from India to U.S. Dollars. There may be a charge applied to the exchange. In particular, given an **amount**> 0 in Rupees, **rate** > 0 is the exchange rate of how much one indian Rupee is worth in dollars and **charge** is the percent charge for the transaction. If there is no charge, then charge is 0.

Write the function `ConvertRupeesToDollars` that has three parameters: `amount, rate` and `charge`, as explained above. This function returns the equivalent amount in dollars minus a charge if there is one.

call	**returns**	**comment**
ConvertRupeesToDollars(1000, 0, .0163)	16.3	no charge
ConvertRupeesToDollars(1000, 20.0, .0163)	13.04	16.3 - 20% of 16.3
ConvertRupeesToDollars(1000, 10.0, .0150)	13.5	15.0 - 10% of 15.0

```
def ConvertRupeesToDollars(amount, charge, rate):
```

B. (8 pts) Consider the problem of calculating the cost of a hotel room for one or more nights. There are four parameters: **price** is the hotel charge for one night, **days** is the number of days to stay at the hotel, **tax** is the percentage tax charge applied to the total price (but not to the fee), and **fee** is an additional tourist fee per night after tax is calculated.

Consider the following rules for computing the cost of a hotel stay.

1. The total price before tax and fees is the charge for one night times the number of days stayed at the hotel.
2. Tax is the percentage charge applied to the total price (but not to the fee)
3. The fee is an extra tourist charge added per night after tax is calculated. However, no tourist fee is applied for a stay of a week or more, and only half of the tourist fee applies for a stay of 4-6 days.

Write the function `hotelStay` below. First, consider the following examples.

call	**returns**	**comment**
hotelStay(100, 1, 10, 20)	130.00	one night for \$100 with 10% tax and \$20 fee = \$100 + \$10 + \$20
hotelStay(100, 8, 10, 20)	880.00	8 nights for \$100/night with 10% tax and no fee = \$800 + \$80 + 0
hotelStay(100, 5, 10, 20)	600.00	5 nights for \$100/night with 10% tax and \$10 fee/night = \$500 + \$50 + \$50

```
def hotelStay(price, days, tax, fee):
```

PROBLEM 3 : (***It's a mystery (16 points)***)

A. (6 pts) Consider the following list named `data` and function `remove` that has two parameters `alist`, which is a list of strings, and `testword`, which is a string.

```
data = ['fruit', 'follow', 'fresh', 'hollow', 'brown']

def remove(alist, testword):
    answer = []
    for word in alist:
        if word[0] == testword[0]:
            if word[1] == testword[1]:
                pass
        else:
            answer = answer + [word]
    return answer
```

This function is suppose to return a new list of the words from `data` that do not start with the first two letters of `testword`, but does not work as intended! Consider the following examples. It doesn't work in the first call, it works in the second call.

call	**returns**	**should return**
remove(data, "freedom")	['hollow', 'brown']	['follow', 'hollow', 'brown']
remove(data, "host")	['fruit', 'follow', 'fresh', 'brown']	['fruit', 'follow', 'fresh', 'brown']

Q1. Give another example of a call to this function with the list data above and a value for testword that does not return the expected value.

```
remove(data,                                )
```

Q2. Explain why this function does not work correctly.

Q3. Here is the code again. Modify the code so it works as intended.

```
def remove(alist, testword):
    answer = []
    for word in alist:
        if word[0] == testword[0]:
            if word[1] == testword[1]:
                pass
        else:
            answer = answer + [word]
    return answer
```

B. (10 pts) Consider the following `mystery` function with two parameters, `people` which is a list of strings, and `key` which is one string.

```
1: def mystery(people, key):
2:     x = []
3:     for item in people:
```

```
4:          if item.find(key) == -1:
5:              x += [item + key]
6:          else:
7:              x += [item]
8:      y = []
9:      for item in x:
10:         if item.find(key)< 4:
11:             y += [item]
12:     return y[0]
```

Consider making the call `mystery(names, "er")` with the value of `names` below. Answer the following questions about tracing what happens with this call

```
names = ['Karl', 'Beth', 'Frederick', 'Sarah', 'Bruce']
```

B1. From the code and call above, list the parameter(s).

B2. From the code and call above, list the argument(s).

B3. What is the value of `x` on line 8?

B4. What is the value of `y` before line 12 executes?

B5. What value is returned from the call `mystery(names, "er")`?

B6. Explain in words the general idea of what `mystery` does.

B7. Give an example of a nonempty list that when passed to `mystery` will crash when run. Explain why it crashes.

PROBLEM 4 : (***Transformations (14 points)***)

PART A (4 pts): Write the function `posUpper` which has one string parameter `word`. This function returns the location of the first uppercase letter in word, or -1 if there are no uppercase letters in word.

```
def posUpper(word):
```

PART B (4 pts): Write the function `posSecondUpper` which has one string parameter `word`. This function returns the location of the second uppercase letter in word, or -1 if there is not a second uppercase letter in word.

YOU MUST CALL `posUpper` that you wrote in part A for full credit.

call	**returns**
posUpper('theIBMer')	3
posUpper('WelcomeAllFriends')	0
posUpper('oHo')	1
posUpper('apple')	-1

call	**returns**
posSecondUpper('theIBMer')	4
posSecondUpper('WelcomeAllFriends')	7
posSecondUpper('oHo')	-1
posSecondUpper('apple')	-1

```
def posSecondUpper(word):
```

PART C (6 pts): Write the function `emphasize` which has one string parameter `word`. This function returns the word transformed in the following way.

1. If the word has just one uppercase letter, return the word with "ly" immediately after the uppercase letter
2. If the word has at least two uppercase letters, return the word with the first uppercase letter doubled and the second uppercase letter with "ly" immediately after that letter
3. Otherwise, return the word with all uppercase letters

YOU MUST CALL `posSecondUpper` that you wrote in Part B for full credit. You may also call `posUpper` that you wrote in Part A.

Note: If you splice a string starting after the last location in the string, it is not an error but returns the empty string.

call	**returns**	comment
emphasize('theIBMer')	'theIIBlyMer'	at least two uppercase letters
emphasize('WelcomeAllFriends')	'WWelcomeAlyllFriends'	at least two uppercase letters
emphasize('oHo')	'oHlyo'	only one uppercase letter
emphasize('apple')	'APPLE'	no uppercase letters

```
def emphasize(word):
```

PROBLEM 5 : (***Info on Basketball Players (20 points)***)

Consider information about basketball players. Assume `data` is a list of strings where each string represents 'lastName#level#school#number' where `lastname` is the last name of the player, `level` is "fr" for first year, "so" for sophomore, "jr" for junior and "sr" for senior, `school` is the school they play for, `number` is the number on their jersey. Assume data has the following value for the examples.

```
data = ['Cook#sr#Duke#2', 'Johnson#jr#UNC#11', 'Freeman#so#NCSU#10',
'TJones#fr#Duke#5', 'Paige#jr#UNC#5', 'Gill#jr#Virginia#13',
'Winslow#fr#Duke#12', 'Towns#fr#Kentucky#12', 'Anderson#jr#Virginia#1',
'MJones#so#Duke#13', 'Ulis#fr#Kentucky#3', 'Hicks#so#UNC#22',
'Brogdon#jr#Virginia#15', 'Okafor#fr#Duke#15', 'Jefferson#jr#Duke#21',
'Booker#fr#Kentucky#1', 'Plumlee#jr#Duke#40', 'Lacey#jr#NCSU#1', 'Turner#sr#NCSU#22']
```

A. (10 pts) Write the function `playersFrom` which has two parameters, `data`, that is a nonempty list of strings in the format above, and `school` which is the name of a school. This function returns a list of the last names of the players from data that are at that school.

call	**returns**
playersFrom(data, "UNC")	['Johnson', 'Paige', 'Hicks']
playersFrom(data, "Duke")	['Cook', 'TJones', 'Winslow', 'MJones', 'Okafor', 'Jefferson', 'Plumlee']

```
def playersFrom(data,school):
```

B. (10 points) Write the function `playersOfTypes` which has three parameters:

1. `data`, that is a list of strings in the format mentioned earlier, where level is "fr" for first year, "so" for sophomore", "jr" for junior and "sr" for senior. The format is 'lastName#level#school#number'
2. `year1` which is a two letter string representing a level
3. `year2` which is a two letter string representing a different level than year1

This function returns the a list of strings in the format 'lastname-school' for those players from data that are level year1 or year2.

```
def playersOfTypes(data,year1,year2):
```

call	**returns**
playersOfTypes(data, ”fr”, ”sr”)	[’Cook-Duke’, ’TJones-Duke’, ’Winslow-Duke’, ’Towns-Kentucky’, ’Ulis-Kentucky’, ’Okafor-Duke’, ’Booker-Kentucky’, ’Turner-NCSU’]
playersOfTypes(data, ”sr”, ”jr”)	[’Cook-Duke’, ’Johnson-UNC’, ’Paige-UNC’, ’Gill-Virginia’, ’Anderson-Virginia’, ’Brogdon-Virginia’, ’Jefferson-Duke’, ’Plumlee-Duke’, ’Lacey-NCSU’, ’Turner-NCSU’]