(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.

```
bo = 4
mo = 2
x = 1.5
print type(mo + x)
print bo - 3 * mo
print bo >= mo
print pow(bo, 2) / 10
print (bo * 2) % 6
```

Output:

**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.

```
city = "Washington"
print city[3]
pos = city.find("g")
print city[pos-4:pos]
print city[-2:]

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

Output:

**PROBLEM 2 : (*The cost of travel - Simple Functions (14 points)*)**

**A.** (6 pts) Consider the problem of calculating the cost of a hotel room for one night. The **price** is the charge for the hotel room (decimal number) , **tax** is a percent charge applied to the hotel charge, and **fee** is an additional tourist fee, a percent charge applied to the hotel price and tax.

Write the function `hotelOneNight` that has three parameters: `price, tax` and `fee`, as explained above. This function calculates the cost of the hotel charge including tax and fee.

```
def hotelOneNight(price, tax, fee):
```

| call | returns | comment |
|---|---|---|
| hotelOneNight(100.00, 10, 10) | 121.00 | 100.00 + 10(tax) + 11 (10% fee on 110) |
| hotelOneNight(100.00, 20, 5) | 126.00 | 100.00 + 20(tax) + 6 (5% fee on 120) |
| hotelOneNight(80.00, 10, 5) | 92.40 | 80.00 + 8(tax) + 4.4 (5% fee on 88) |

**B.** (8 pts) Consider the cost of flying a group of people together on one-way tickets. There are four parameters: **price** is the regular cost of a one-way airline ticket, **day** is the day of the week they fly, **tax** is the percentage tax charge applied to the total price after all discounts are taken, and **people** is the number of people flying.

Consider the following rules in calculating the total price for the group.

1. If the group flys on Tuesday or Wednesday, there is a discount of $5 off for every $100 spent on the total cost for the group.
2. After the day discount is taken if it applies, there is a discount of $20 for each person over two people in the group.
3. The tax is applied after all discounts are taken.

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

| call | returns | comment |
|---|---|---|
| travelCost(110.00, "Monday", 10, 3 ) | 341 | 3 people - 20 for one, plus 10% tax = ($330 - $20) + 31 |
| travelCost(110.00, "Wednesday", 10, 2 ) | 231 | 2 people - day discount, plus 10% tax = ($220 - ($5 * 2)) + 21 |
| travelCost(110.00, "Tuesday", 10, 4 ) | 418 | 4 people - day discount, plus 10% tax = ($440 - ($5 * 4) - $20 * 2) + 38 |

```
def travelCost(price, day, tax, people):
```

**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.

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

```
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 |
|---|---|
| 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 `numbersFrom` 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 strings of the numbers of the players from data that are at that school.**

| **call** | **returns** |
|---|---|
| numbersFrom(data, "UNC") | ['11', '5', '22'] |
| numbersFrom(data, "Duke") | ['2', '5', '12', '13', '15', '21', '40'] |

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

**B.** (10 points) Write the function `playersOfTypeAndNumber` 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. `year` which is a two letter string representing a level
3. `number` which is an integer number (NOT A STRING)

**This function returns the list of strings in the format 'school-lastname' for those players from data that are level year and have a player number that is less than number.**

| **call** | **returns** |
|---|---|
| playersOfTypeAndNumber(data,"fr",12) | ['Duke-TJones', 'Kentucky-Ulis', 'Kentucky-Booker'] |
| playersOfTypeAndNumber(data,"sr",25) | ['Duke-Cook', 'NCSU-Turner'] |

```
def playersOfTypeAndNumber(data,year,number):
```