CS 116 Winter 2013

Assignment 06
Due at 10:00 am on Wednesday, March 13

- All solutions are to be in Python.
- Do NOT use Python iteration (loops). Any repetition should be implemented using recursion or abstract list functions.
- You may import and use the math module if needed for any question
- Download the testing module from the course webpage. Include `import check` in each solution file.
- Read and follow the instructions in the Python Style Guide for CS116 available on the course Web page.
- You are encouraged to use helper functions in your solutions as needed. Include them in the same file as your solution but make helper functions separate functions from the main function, i.e., do NOT make them local functions. You do not need to provide examples and tests for these helper functions.
- Be sure to use strings that exactly match those specified on the assignment and the interface. Changing them in any way may result in the loss of correctness marks.
- You are not allowed to use global variables
- Do not copy the purpose directly from the assignment description. The purpose should be written in your own words and include reference to the parameter names of your functions.
- The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources. Do not email or share your code with any of your fellow students.
- Read the course Web page for more information on assignment policies and how to organize and submit your work. Follow the instructions in the style guide. Specifically, your solutions should be placed in files a06qY.py, where Y is a value from 1 to 4.
- Download the interface file from the course Web page.
- For full marks, it is not sufficient to have a correct program. Be sure to follow all the steps of the design recipe, including the definition of constants and helper functions where appropriate.
- Read each question carefully for restrictions before posting on piazza. Test data for all questions will always meet the stated assumptions for consumed values.
- Assignments will not be accepted through email. Course staff will not debug code emailed to them.

Coverage: Module 6

Language: Python

1. Write a function *`find_email`*, which consumes nothing and produces None. The function will prompt the user to input a string that contains the email message and a string that needs to be searched in the email message. The function prints the number of occurrences of the string appearing in the email message. The function should be able to find all the occurrences in case of overlapping strings.

For example: ***`Calling find_email() prompts the user`***

```
Enter the email message: This is final result of the final examinations
Enter the word to find: final

Prints
2

find_email() prompts the user
Enter the email message: Is anna a banana
Enter the word to find: ana
```

```
Prints
2

find_email() prompts the user
Enter the email message: aaaaaa
Enter the word to find: aa

Prints
5
```

2. Write a function `student_grade`, which consumes two lists `full_names` and `percentages` both containing the same number of elements and produces a list of lists. The elements in the list `full_names` are of type string containing student full names and the elements in the list `percentages` are of type integers with the values between [0-100], containing student percentages corresponding to the students in the list `full_names`. The function checks the student percentages and assigns the appropriate letter grade to each student based on the following table:

Percentage	Grade
80-100	"A"
70-79	"B"
60-69	"C"
50-59	"D"
0-49	"E"

The function produces a list of lists, the same length as `full_names`. Each list in the produced list is of length 3, and contains, in order, the `student_name (a string)`, `student percentage (an integer)` and a letter `grade (a string)`, or an empty list if both `full_names` and `percentages` are empty lists.

For example

```
student_grade (["Troy" , "Lori" , "Adriel", "Rosina" , "Maheen" ,"Dan", "EDI"]
, [89,84,74,72,61,42,52]) =>
[['Troy', 89, 'A'], ['Lori', 84, 'A'], ['Adriel', 74, 'B'], ['Rosina', 72,
'B'], ['Maheen', 61, 'C'], ['Dan', 42, 'E'], ['EDI', 52, 'D']]

student_grade ([] , []) =>[]
```

3. Consider a dice game called *BHCS*. It is multi-player game where each player is given four dice. Each die has only one of four possible faces **(bear, hat, cupcake, shoe).** Each face earns a certain number of points as described below.

Each player rolls all four dice in each turn and the winner is the player with maximum number of points. The points for each face are

bear = 10 points

hat = 5 points

cupcake = 2 points

shoe = 0 points

Write a function *bhcs_game* that consumes a non-empty list of lists of strings (*die_rolls)* and produces a string that indicates the outcome of the game. Each list in *die_rolls* should contain four strings representing the face for all of the dice for each player (i.e. each list of length 4, which is one player's turn). The produced string contains the number of the winning player along with the player's point total, in the following format:

Player XX is the winner with YY points

where XX is the number of the winning player (player 1 is at position at 0 in die_rolls) and YY is their point total.

In case of a tie the function produces a string

There is a tie

For example:

bhcs_game([["cupcake", "hat", "shoe", "bear"]] => 'Player 1 is the winner with 17 points'

bhcs_game([["shoe", "hat", "hat", "cupcake"],["cupcake", "hat", "shoe", "bear"]])=> 'Player 2 is the winner with 17 points'

bhcs_game ([["cupcake", "hat", "shoe", "bear"],["cupcake", "hat", "shoe", "bear"],["hat","cupcake" ,"bear","shoe"]])=> 'There is a tie'

4. Rogers home phone wants to implement an ***Electronic Telephone Directory*** **[ETD]** for their business convenience. The ***ETD*** provides the basic functionality of adding new records, deleting records and viewing records. You will implement this program to aid in this task. The addition of a new record, or the deletion of an existing record is based on the phone number. In order to add a record the *phone* of the new record is checked with all the existing records and customer will be added only if the *phone* appears as a customer phone number. Similarly a record is deleted only if *phone* appears as a customer phone number.

Consider the following data definitions

customer_record is a list of length 3: *[name, address, phone],* where *name* is a non empty string, *address* is a non empty string and *phone* is a 10-digit integer

Assignment 06
Due at 10:00 am on Wednesday, March 13

`customer_list` is a list of `customer_record`

Create the following functions to implement the required functionality where `cust_record` is of type `customer_record`, `all_records` is of type `customer_list` and `phone` is a 10-digit integer. You are required to use the Abstract List Functions and list methods for this question. You must **not** use recursion for this question.

• ***`add_record:`*** Adds a new record to the list of existing records. The function consumes the list `all_records` and `cust_record` and produces None. The function mutates the list `all_records` by adding a new record `cust_record` at the end if there is no duplicate record, or prints the following message if the record already exists

`Phone number XXX is already in the existing customer records`

where XXX is the phone number of the customer being added

• ***`delete_record:`*** Deletes the record from the list of existing records based on phone number. The function consumes `all_records` and `phone,` and produces None. The function mutates the list `all_records` by deleting a record with matching `phone,` or prints the following message if there is no customer record with the matching `phone`

`No record found with phone number XXXX`

where XXX is the phone number of the customer being deleted

• ***`view_records:`*** Produces a list of records that matches the phone number in the manner described below. The function consumes `all_records` and `phone`, and produces a list of `customer_record` values that matches the `phone` or an empty list if there is no matching record. If there is a typing mistake in the `phone` for at most one number at the start or end then the function should produce all the records with the `phone` numbers having at least nine consecutive identical digits in common with the `phone` digits

For example:

Suppose
```
all_records = [["Cust1", "Kitchener", 5192345647]]
```

calling
```
add_record(all_records,["Cust2", "Cambridge", 5193243456])=>None
```

and mutates
```
all_records =[['Cust1', 'Kitchener', 5192345647], ['Cust2', 'Cambridge',
5193243456]]
```

Suppose
all_records= [["Cust1", "Kitchener",5192345647]]

Calling
add_record(all_records,["Cust2","Cambridge", 5192345647])=> None

prints
Phone number 5192345647 is already in the existing customer records.

Note: *all_records* has not been changed in this example

Suppose
all_records=[["Cust1","Kitchener", 5192345647],["Cust2","Cambridge", 5193243456]]

Calling
delete_record(all_records,5192345647)=> None

and mutates
all_records=[['Cust2','Cambridge',5193243456]]

Suppose
all_records[["Cust1","Kitchener",5192345675],["Cust2","Cambridge", 5193243456]]

calling
delete_record(all_records,5192345643)=> None

prints
No record found with phone number 5192345643

Note: *all_records* has not been changed in this example

Suppose
all_records

=[["Cust1","Kitchener",5192345647],["Cust2","Cambridge", 5193243456],["Cust3","Waterloo",5192345648],["Cust4","Guelph", 5192344589]]

calling
view_records(all_records,5192345648)
=> [['Cust3', 'Waterloo', 5192345648]]

Assignment 06

Due at 10:00 am on Wednesday, March 13

suppose

```
all_records= [["Cust1","Kitchener",5192345647],["Cust2","Cambridge",
5193243456],["Cust3","Waterloo",5192345648],["Cust4","Guelph",5192343527]]
```

calling

```
view_records(all_records,5192345649)
=> [['Cust1', 'Kitchener', 5192345647], ['Cust3', 'Waterloo', 5192345648]]
```

suppose

```
all_records=[["Cust1","Kitchener",5192345647],["Cust2","Cambridge",
5193243456],["Cust3","Waterloo",5192345648],["Cust4","Guelph",5192344589]]
```

calling

```
view_records(all_records,5199875645)

=> []
```