**Week Extra 11 In-Class Exercises (dictionaries)**

**Q1: Seating Plan**

You are given a file called “seating\_plan.txt” that contains the seating arrangement for a conference. The seats in the conference hall are labelled with a row letter and a seat number. Row letters are 'A', 'B', 'C', etc. Seat numbers are 1, 2, 3, etc. Each line of the file contains three columns: (1) A row letter. (2) A seat number. (3) The name of the person to be seated at that row and seat. The three columns are separated by commas.

**Part (a) [ \*\*\* ]**

Read the file and store it in a dictionary. Prompt the user for a row letter and a column number. Display the person to be seated at that seat or display “Seat not taken” if nobody has been assigned to sit there.

Hint: Think about what should be used as keys in order for you to quickly find the person sitting at a specified seat. Note that keys can be tuples.

Three sample runs of the program can be found below:

* Sample Run #1:

Enter a letter (from A to E) :A

Enter a seat number (from 1 to 25) :1

The person seated at the seat A1 is Michelle Lee

* Sample Run #2:

Enter a letter (from A to E) :B

Enter a seat number (from 1 to 25) :21

The person seated at the seat B21 is Lindy Chan

* Sample Run #3:

Enter a letter (from A to E) :C

Enter a seat number (from 1 to 25) :15

Seat not taken

**Part (b) [ \*\*\* ]**

Modify the program in Part (a) such that if the seat is taken by a person, after displaying that person’s name, also try to find who sits beside him/her, i.e., the person sitting on the left and the person sitting on the right of this person.

Three sample runs of the program can be found below:

* Sample Run #1:

Enter a letter (from A to E) :B

Enter a seat number (from 1 to 25) :20

The person seated at the seat B20 is Eric Wong

The person seating on the left side of that person is Ng Yin Hui

The person seating on the right side of that person is Lindy Chan

* Sample Run #2:

Enter a letter (from A to E) :A

Enter a seat number (from 1 to 25) :12

The person seated at the seat A12 is Wendy Li

Seat not taken on the left side

The person seating on the right side of that person is Chua Lee Hong

* Sample Run #3:

Enter a letter (from A to E) :A

Enter a seat number (from 1 to 25) :13

The person seated at the seat A13 is Chua Lee Hong

The person seating on the left side of that person is Wendy Li

Seat not taken on the right side

**Q2: Degrees of Separation [ \*\*\*\* ]**

You might have heard of “six degrees of separation” (<https://en.wikipedia.org/wiki/Six_degrees_of_separation>). Indeed, we live in a small world.

You are given a file called “friends.txt” that stores people who are friends with each other. Each line of the file contains two people’s names separated by a tab. These two people are friends with each other.

**Part (a)**

Write a program that does the following. The program prompts the user for a person’s name. Call this person **X**. The program then prompts the user for an integer between 1 and 6. Call this number **n**. The program then displays all people who are within **n**-degrees of separation from **X**.

You can assume that **X** has at least one friend, i.e., **X** has appeared in the file “friends.txt.”

For example, suppose the file contains the following data:

A B  
A C  
A D  
B C  
B E  
C D  
C F  
D G  
F G

* Example #1: If **X** is E and **n** is 1, then the program should display only B because 1-degree separation means direct friends.
* Example #2: If **X** is E and **n** is 2, then the program should display A, B and C. This is because B is 1-degree separated from E, while A and C are 2-degree separated from E.
* Example #3: If **X** is E and **n** is 3, then the program should display A, B, C, D and F. This is because in addition to A, B and C, now D and F are 3-degree separated from E, and therefore should be added.

**Part (b)**

Change the program above so that when **X** and **n** are given, the program prints out those people who are exactly **n**-degree separated from **X**.

**Part (c)**

Write a program that prompts for two names. The program then displays how many degrees away these people are. (You can assume that these two people are connected.)

* Example #1: Given the file above, if the two people are A and E, then the program should display **2**, because A is connected to B and B is connect to E. (Note that although A could also be connected to E through A 🡪 C 🡪 B 🡪 E, this is a longer path, so we do not consider this path.)
* Example #2: If the two people are B and G, then the program should display 3.