Timed Lab 1 – Contact Book

This is a timed lab; this Timed Lab is worth 25 **Exam** points.

For this Timed Lab, you may use:	However, you **may not**:
• Course notes • Homeworks • Recitation assignments • Other course material • Any material you may find on the Internet that doesn't involve communicating "live" with other people	• Communicate with other people/students in real-time via any means. This means no Facebook, email, Piazza, IM, IRC, cell phones, Google Talk, smoke signals, writing messages in the database, etc. • Share code with other students • Look at other student's work

The TAs will be available to answer clarifying questions about the problem, but they are not permitted to give assistant with debugging code, program logic, etc. You will have an entire recitation period to work on this assignment; this time begins *exactly* when your recitation begins and ends *exactly* when your recitation ends. No extra time will be given if you arrive late, except in highly extenuating circumstances that must be approved by Dr. Summet.

T-Square will not permit any late submissions; ensure that you submit your code to T-Square several times to prevent earning a zero due to you being unable to submit. Your TAs will give a verbal warming at 10 and 5 minutes before the end of a recitation period; you should submit at these times. In addition, before leaving your recitation section, be sure to have received the confirmation email of your T-Square submission!

In your collaboration statement, if you use code from somewhere that is **not** a class resources (i.e., not listed on the course calendar), please list where this code came from. Ensure that you fill out the header at the top of the tile.

Note that you must check out with your TA before you leave the recitation. If you do not check out with your TA or you modify your submission after you leave the recitation room, **you will receive a zero on the timed lab.** No submission will be accepted after T-Square closes the assignment (i.e., it won't let you submit).

Problem Description:

You have not quite yet adapted to the times and have a physical contact book, storing your friends' contact information. Now that you're proficient in Python, you've decided to create a program that will help you create, update, and display contact information.

The contact book will be a **global** dictionary named ContactsDict. You must define and use the same dictionary throughout all of your code. Each time you restart your code, the dictionary will clear. However, the dictionary must be updated accordingly for all function calls.

Required Functions:

createContact – This function takes in two parameters, the name of the contact to be created as a string and a list of contact information; there is no return value. Using this input, you will create a

dictionary entry for the given contact. The street and city will be strings, and the phone number an integer. You may assume that each name is unique. The key of the dictionary will be the name, and the value will be the information formatted in the following way:

[(street, city) , phoneNumber]

updateContact – Now that you've found a way to create your contact book, you realized that the information for your friends is constantly changing; this function will help you stay up to date! The function takes in no parameters and has no return value. The function must:

- Ask the user for the name of the contact being updated
- Ask the user which part of the contact information is changing (you may assume only one piece of information is changing)
- Update the dictionary appropriately

The function should be able to handle invalid inputs (i.e. names that don't exist, incorrect spelling, etc). If there is an error at any point, the function should print "Invalid input!" and restart.

printContacts – You have these lovely Python functions that enable you to store your contact book, but it isn't easy to retrieve information from. This function has one parameter, the number of contacts to be printed to the shell. The contacts should always be printed in alphabetical order. If the number passed through the function is larger than the number of contacts, the entire contact book should be printed. The contact book should be printed in the following format:

Name
    City
    Street
    Phone Number
Name
    …

Additional Notes:

createContacts -

'info' parameter will be given in following format: [street, city, phone]

updateContact -
    The name cannot be changed
    There should be THREE prompts
    -name of contact being changed
    -which part of the contact information is being changed
    -what the new information is
    Be specific with prompts
    i.e. "What are you changing? Enter 'street', 'city', or 'phone':"
    Unchanged contact information should be present in the updated dictionary entry

Grading:

You will earn points for each piece of functionality that works correctly according to the specifications.

createConctact:		**7**
Correct function header	2	
Creates a dictionary entry	2	
Dictionary entry correctly formatted	3	
updateContact:		**9**
Prompts user for the name	1	
Prompts user for the changing part	1	
Prompts user for new information	1	
Attempts to update dictionary entry	2	
Function can correctly change phone	1	
Function can correctly change street	1	
Function can correctly change city	1	
Invalid input handled correctly	1	
printContacts:		**9**
Correct function header	1	
Attempts to iterate through keys of dictionary	2	
Has a print statement	1	
Prints any number of contacts in any order	1	
Prints correct number of contacts	2	
Prints contacts in alphabetical order	1	
Can handle any input number without error	1	