Lab exercise Week 6
Q1
You are asked to write a program that reads a file with accounts then allows you to processes some accounts by name and adding amounts. You will need to catch exceptions and handle them.
To do this, you need to write two functions: **`readAccounts(infile)`** to read the account file, and **`processAccounts(accounts)`** to make changes to accounts.

In the main function of your program the user will be asked to enter the name of the file to read. If the file does not exist, the main function should be able to catch the IOError exception. Upon catching the exception, the main function should print an error message and exit the program gracefully (i.e. no crash with call stack displayed). Here is a sample run that shows what happens when the filename entered by the user does not exist:

```
Enter filename> sillyfile.txt
IOError. Sillyfile.txt does not exist
```

If the file exists, the main function opens the file in read mode and proceeds to call the function readAccounts(). The main function passes the TextIOWrapper object (the file handler) as an argument to this function.

Each line in the file is a record of a person's name followed by a colon symbol followed by a number that represents the amount the person has in their account. The function `readAccount` reads the file one line at a time and stores the `name:account` as a key:value pair inside a dictionary. If the amount value associated to a name is not a valid number the function should raise an exception and that `name:amount` pair with the invalid amount, is not added to the dictionary. If an exception is raised, the following message is printed by the function:

```
ValueError. Account for Smith not added: illegal value for balance
```

After raising and catching the exception the program should continue to the next record in the file.

Once all records with valid amounts have been added into the dictionary, the `readAccounts` function should return this dictionary.

The main function should now call the `processAccounts()` function and pass the dictionary that was returned by `readAccounts()` as an argument. The `processAccounts()` function should ask the user to enter the name of a person. If the entered name does not exist as a key in the dictionary, a KeyValue Error is

raised and should be caught by the program. For example is AliBaba is given as a name and these no such key in the dictionary, the following message is displayed :

```
KeyError. Account for AliBaba does not Exist. Transaction
cancelled.
```

After catching the exception and handling it, the function asks the user to enter name again. If the name exists in the dictionary, the function asks the user to enter the amount value that would be added to the existing amount associated to the name in the dictionary. If the user enters a value that is not a float or integer, the function raises and catches a ValueError exception and the following message is displayed:

```
Value Error. Incorrect Amount. Transaction cancelled.
```

After catching and handling the exception, the program allows the user to continue entering and new name of an account holder and the transaction amount until the user enters the word “Stop”. If the name exists in the accounts dictionary and the amount is a valid integer or float value, the transaction goes through and the function reports the amount in the account after been updated. Please note you don’t have to update or write to original file. You just need to update the amount in the dictionary. **Here is a complete sample run of the program:**

```
Enter filename >accounts.txt
ValueError. Account for Smith not added: Illegal value for balance
ValueError. Account for George not added: Illegal value for balance
Enter account name, or 'Stop' to exit: bob
KeyError: Account for bob soes not Exist. Transaction cancelled.
Enter account name, or 'Stop' to exit: Bob
Enter transaction amount for Bob: 30
New balance for account Bob: 264.7
Enter account name, or 'Stop' to exit: Zoya
Enter transaction amount for Bob: hello
Value Error. Incorrect Amount. Transaction cancelled.
Enter account name, or 'Stop' to exit: Johnson
Enter transaction amount for Johnson: -20
New balance for account Bob: 771.56
Enter account name, or 'Stop' to exit: Stop
```

Q2. In this program, the user is supposed to enter a date in YYYY-MM-DD format. You are required to create a function named `isValidDate()` with parameters being the year, month and day extracted from the user input. This function uses assertions to check that the year, the month and day are correct. The month should be an integer between 1 and 12. The day should be an integer valid for the days of the corresponding month, up to 29 for February. To keep is simple, let’s say the

year should be between 1901 and 2020 and the leap year are years divisible by 4. In general, every year that is exactly divisible by four is a leap year, except for years that are exactly divisible by 100, but these centurial years are leap years if they are exactly divisible by 400. For example, the years 1700, 1800, and 1900 are not leap years, but the year 2000 is.