CPE 101 Spring 2017
Program 5-Due 6/9/17 for Full Credit
*(File Matching) F*ile-processing program

Objectives:

- More practice on 2D List in Python,
- Practice on sorting a 2D List in Python,
- Practice on reading from file in Python.
- Practice on writing to a file in Python.

Due Date:

Friday (5/9) by 11:55PM for 100%
Saturday (6/10) by 11:55PM for 90%
Sunday (6/11) by 11:55PM for 80%

Required File Header:

All students are required to have the following header comment at the top of their source files.

```
# Project 5 – File Matching #
# Name: Your Name
# Section: Your Section
# Instructor: Your Instructor Name
```

Resources:

- Your text.
- Your instructor and TA.
- Tutoring Center (Sun-Thur, 7:00-9:00pm)

What do you need to submit:

- Functions definitions for your program, fileMatchingFuncs.py
- Testing script which calls your functions, funcsTests.py.
- Output files, which are sorted "sorted_oldMaster.dat" and a new master file, "newMaster.dat"

Ground Rules:

- Your program must read two input files
- Your program must generate two output files

Program Description:

In commercial data processing, it's common to have several files in each system. In an accounts receivable system, for example, there is generally a master file containing detailed information about each customer such as the customer's name, address, telephone number, outstanding balance, credit limit, discount terms, contract arrangements and possibly a condensed history of recent purchases and cash payments. In this program, we only stored the customers' account number, customers' first and last name, customer's balance, customers' phone number, and customers' city.

As transactions occur (i.e., sales are made and cash payments arrive in the mail), they're entered into a file. At the end of each business period (i.e., a month for some companies, a week for others and a day in some cases) the file of transactions (called "transaction.dat") is applied to the master file (called "oldMaster.dat"), thus updating each account's record of purchases and payments. After each of these updates, the master file is rewritten as a new file ("newMaster.dat"), which is then used at the end of the next business period to begin the updating process again.

File-matching programs must deal with certain problems that do not exist in single-file programs. For example, a match does not always occur. A customer on the master file might not have made any purchases or cash payments in the current business period, and therefore no record for this customer will appear on the transaction file. Similarly, a customer who did make some purchases or cash payments might have just moved to this community and the company may not have had a chance to create a master record for this customer.

Use the account number on each file as the record key for matching purposes. The oldMaster.dat is not ordered. You need to read the file and generate a sequential file which is

sorted in increasing account number order. This sequential file will be sorted_oldMaster.dat. The transaction.dat file has records of account numbers and value.

When a match occurs (i.e., records with the same account number appear on both the master file and the transaction file), add the dollar amount on the transaction file to the current balance on the master file and write the "newMaster.dat" record. (Assume that purchases are indicated by positive amounts on the transaction file, and that payments are indicated by negative amounts.) When there is a master record for a particular account but no corresponding transaction record, merely write the master record to "newMaster.dat". When there is a transaction record but no corresponding master record, print the message "Unmatched transaction record for account number …" (fill in the account number from the transaction record)

For example:

Unmatched transaction record for account 900

The data files and sample of output files can be found in the PolyLearn. Copy the files in your working directory in Linux.

Submission:

1. fileMatchingFuncs.py,
2. fileMatching.py,
3. funcsTests.py,
4. sorted_oldMaster.dat, and
5. newMaster.dat.

Sample Files:

Given files:

1) oldMaster.dat

```
100  Alan  Jones      348.17     8053564820      SLO
700  Suzy  Green      -14.22     8052586912      SLO
300  Mary  Smith      27.19      8057901237      Santa_Maria
800  Mike  Rosen      -104.58    8051200891      Pismo_Beach
```

2) transaction.dat

```
100   27.14
300   62.11
400   100.56
700   100.0
900   200.0
```

Generated files:

1) sorted_oldMaster.dat

```
100  Alan  Jones      348.17     8053564820      SLO
300  Mary  Smith      27.19      8057901237      Santa_Maria
700  Suzy  Green      -14.22     8052586912      SLO
800  Mike  Rosen      -104.58    8051200891      Pismo_Beach
```

2) newMaster.dat

```
100  Alan  Jones      375.31     8053564820      SLO
300  Mary  Smith      89.30      8057901237      Santa_Maria
700  Suzy  Green      95.78      8052586912      SLO
800  Mike  Rosen      -104.58    8051200891      Pismo_Beach
Unmatched transaction record for account 900
```

Note:

Based on different input files you will create new version of sorted_oldMaster.dat and newMaster.dat files. You can create sequence of these files by putting a sequence number or over write the old one.