Adap.tv - Programing Challenge

The Dogyard Book store assigns a unique code to books they stock which is 'P' followed by a number - for instance, '*The Hitchhiker's Guide to the Galaxy*' is listed as **P42** in their price list catalog. Each book has an entry in the file as a comma-separated list - book-id, price, bookname. A sample price list (`price.list`):

```
P342,500,Black Holes and Baby Universes
P8,90,Love in the time of Cholera
P675,23,Number Theory and Cryptography
P563,1000,Lord of the Rings - Box Set
P456,12,Da Vinci Code
P546,20,Linux Device Drivers
P673,45,The Great Indian Novel
P1,34,Predictably Irrational
P42,44,The Hitchhiker's Guide to the Galaxy
P99,99,Problems in Physics
```

Dogyard gives each of their customers a card with a unique customer-id which is 'C' followed by a number - for instance - **C45723**. They record each purchase that a customer makes as a row of comma separated values - a customer-id followed by the book-ids of the books he/she purchased as part of that transaction. Thus, a customer may have one or more transactions in the transaction file. A sample transaction list(`transaction.list`):

```
C12397,P342,P8,P563,P456
C3452,P546,P8,P673,P675
C1238,P1
C12397,P8,P673,P42
```

The management at Dogyard wants to do a better job at their marketing and inventory planning effort. Your task is to write a program that does the following operations:

1. Generate a report that has:

the top N frequent customers and the number of visits they made

the top N highest transactions and the corresponding customerid

the top N highest selling books and quantity sold

the N least selling books and quantity sold (this includes books that did not sell at all as well)

In each of the cases above, if there is a tie, output all eligible customers/books.

2. Also, as part of their rewards program Dogyard would like to offer a 10% discount to all customers whose sum total of purchases exceed a certain value V. Given a customer-id, your program should be able to determine if this customer is eligible for a discount or not.

The program should be able to accept the following parameters as arguments:

```
-t <path/to/transaction/file>
-p <path/to/price/list/file>
-r <N> ; N is a number greater than ZERO for generating the report
-d <V> -c <customerID> ; prints a 1 if the customer is eligible or 0 if the customer is
not eligible
```

and produce results to standard output. Given the sample files listed above, the program 'yoda' produces the following output:

Output

```
$./yoda -t transaction.list -p price.list -r 3
C12397 2,C1238 1,C3452 1,
C12397 1602,C12397 179,C3452 178,
P8 3,P673 2,P1 1,P342 1,P42 1,P456 1,P546 1,P563 1,P675 1,
P99 0,P1 1,P342 1,P42 1,P456 1,P546 1,P563 1,P675 1,P673 2,

$./yoda -t transaction.list -p price.list -r 2
C12397 2,C1238 1,C3452 1,
C12397 1602,C12397 179,
P8 3,P673 2,
P99 0,P1 1,P342 1,P42 1,P456 1,P546 1,P563 1,P675 1,

$./yoda -t transaction.list -p price.list -r 1
C12397 2,
C12397 1602,
P8 3,
P99 0,
```

When there is a tie, i.e. there are multiple matching candidates, sort in increasing order and print them. The example output above illustrates this for frequent customers when r=2. C1238, C3452 both have visited once and are output in the lexical order of the customer id - **C1238 1,C3452 1**. Similarly, for the most and least sold books when r=2 or r=3. Books P342, P42, P456, P546, P563, P675 and P1 all sold one copy each but the output has them in increasing lexical sorted order as **P1 1,P342 1,P42 1,P456 1,P546 1,P563 1,P675 1,**

```
$./yoda -t transactions.txt -p price.list -d 1700 -c C12397
1
$./yoda -t transactions.txt -p price.list -d 1700 -c C1238
0
$./yoda -t transactions.txt -p price.list -d 150 -c C3452
1
```

DIRECTIONS:

1. Please ensure that your program can be compiled by gcc/g++/java and executed on a Linux machine. It is encouraged that you write a makefile to compile your program. Do not send just the executables. We want to see your program as well. Also include a description of your design/approach to the solution.
2. Your program must be fast, efficient and, most importantly, produce the correct output. Credit if you have taken care of error handling. Extra credit if you can show us some really interesting aspect of software design or coding with your solution.
3. You have 3 hours to solve the problem, test and submit your solution. Your goal is to provide a working solution at least. You may submit additional code later if you think you can make it work better or faster.
4. **A script will test your program across a varied set of inputs for correctness. Please adhere to the output format - values are separated by a space and then pairs of values by commas - <key1><SPACE><value1>,<key2><SPACE><value2>, …,<keyX><SPACE><valueX> . The output of your program must at least match the sample output given above generated by using the sample input files.**

Hint: C/C++ command line options may be parsed using - http://www.gnu.org/software/libc/manual/html_node/Getopt.html . A GNU java port is also available - http://www.gnu.org/software/gnuprologjava/api/gnu/getopt/Getopt.html

In case you have trouble, please feel free to reach out to us over email and we'd be happy to help you out with doubts and clarifications.