Computer Science II — Homework 7 — CS2Flix

This assignment is due Thursday, April 5th at 11:59:59pm and is worth 125 points toward your homework grade. You may use any technique covered thus far this semester, but you must demonstrate the use of at least one map. **Warning:** this assignment is one of the more difficult of the semester, so use your time wisely. You must submit your source code files and a readme.txt file combined into a zip file.

The CS2Flix company has hired you to develop a software system to manage their customers and their movie inventory. Armed with your new found knowledge of maps, you agree to tackle this problem. You agree to do it for free because you want a good grade on Homework 7 and you want to establish your reputation as an outstanding programmer.

Your program should have two input files, one that describes the movies and one that describes a set of rental and return requests. All output should go to a third file.

The movie file contains movie names and the id numbers of individual copies of the movies. Each movie is specified on a single line of input and its id numbers, each an individual string, are specified together on the next line of input. You must convert the input line for a movie to a single string representing the title. For example,

```
Live and Let Die
hadf68e43 8s7df346  9dfer934  ku876df
```

indicates the movie `"Live and Let Die"` with four separate copies. When reading this input you must allow for minor variations in the way the title of the movie is described. In particular, you must remove all leading and trailing whitespace characters, before and after the moving name, and you must remove intervening white-space that is more than a single `' '`. You do not need to remove punctuation or convert to lower case. Thus,

```
Live    and   Let   Die
```

is the same as

```
Live and Let Die
```

but different from

```
Live And Let die
```

There are no blank lines in the movie file. You should be able to parse the movie file based on what you have learned about string and character operations.

The rental/return request file is organized day-by-day. For simplicity, days are non-negative integers, and a change in days is specificed by the input string `"next-day"`, which will be on a line of input by itself. There are only two other types of input in this file

```
rent customer-id movie-name
```

and

```
return customer-id movie-name
```

In these, `customer-id` is a single string, while `movie-name` is in the same form as in the movie file.

Here is how to handle the input. Each customer is allowed to rent up to 3 movies for up to 5 days each and may not rent any movies if the s/he has a movie that is more than 4 days late (more than 9 days after the movie was rented). Therefore, when a customer requests a movie through the `rent` input, several things must happen:

- If the customer is unknown, then add the customer. Output a message when a customer is added.
- If the customer is not allowed to rent any more movies, then a message should be output to this effect. Check first if the customer already has three movies, and then only if this check passes check to see if the customer has a movie that is more than 5 days late.
- If the customer already has that particular movie, do not allow the customer to rent a second copy.
- If the customer is allowed to rent a movie, proceed to checking the availability of the particular movie requested. If the movie is not in the set of movies then the system should output a message saying that the movie is unknown.
- If all copies of the movie have been rented then output a message saying that the movie is unavailable.
- Finally, if everything is ok, then rent the movie to the customer. To keep things consistent, rent the currently-available copy that has the lowest id in lexicographic order.

The `return` should be handled quite simply by recording the fact that the copy of the movie is now available and outputing a message. You may assume that the movie and the customer-id are both correct — the automatic label reader does not make any errors when movies are returned.

The program should start with day 0 and increment a counter that indicates the current day whenever the input request `next-day` is seen in the input and should output a blank line followed by

```
******
Day N
******
```

where N is the current day (N starts with 0). In addition, each time `next-day` is read from the input, the program should output the names of the customers and the movies held by the customer that are late. This output should indicate the number of days late for each movie. Both the customers should be in alphabetical order and the movies should be in order of rental.

At the end of the input your program should output a blank line, and then

```
Summary
-------
```

Then, it should output the customer-ids, in lexicographic order, and for each id, the list of movies rented by the customer, in the order they were first rented. The movies should be output one-per-line and indented by a tab (`'\t'`). If a customer rented no movies then output `<none>` instead of the movie. If a customer rented a movie more than once, only output the movie title once.

See the examples posted on-line for input / output formats and other clarifications.

The command-line should run as

```
cs2flix movies requests results
```

where `movies` and `requests` are the input files and `results` is the single output file.

Final notes:

- Your program must demonstrate the use of maps. You may find maps useful in more than one place in this assignment.
- No structs may be used and no public member variables may be used in your classes.
- Of course, no global variables may be used, but global constants indicating the number of movies allowed per customer, the number of rental days, etc. are not only allowed, but are good programming practice.