Linked List and Lotteries: Input, Output, and Data Description

Haverford CS 106 - Introduction to Data Structures

Lab 3 (due March 19th)

1 Overview

This document describes the data, input, and output for Lab3 besides providing guidelines on the use of command line input, and instruction to work with comma-separated value (CSV) files.

2 The Input

2.1 The Command Line

2.1.1 Syntax

- The command line arguments will be passed in the following format:

```
<registration.csv> [--find <studentName> <courseName>]...
[--print <courseName> {all | enrolled | waitlist}]...
```

- The first argument is a file name. The following two arguments are optional flags. Each optional flag will be followed by two arguments, and the second argument for the `print` flag will only be `all`, `enrolled`, or `waitlist`. You can assume that any given command line argument will perfectly follow this syntax; i.e. there will be no "bad" input with our test cases.

- For the basic (no extensions) lab work, the arguments after `print` flag and the `courseName` argument after the `find` flag can be ignored.

2.1.2 Examples

Following are some examples command lines input which follow the aformentioned syntax.

- `cs106-registration.csv --find Lizzie CS106`
- `students.csv --find Blien CS106 --print CS106 enrolled --print CS106 waitlist --find Steve CS106`
- `registration.csv --print CS231 all --find Lee CS231`
- `registration.csv`

For the basic (no extensions) lab, we will only test one flag per command line.

2.1.3 Description of the Syntax

Per Microsoft's syntax key (various others tend to be similar), the notations can be understood as follows:

- Text not in angle brackets should be typed as shown.
- The angle brackets `<>` denote placeholders that should be substituted with described arguments.
- The square brackets `[]` denote optional items that may or may not be present in the syntax. They do not necessarily come in order.
- The curly braces `{}` denote a set of mutually exclusive items, separated by vertical bar `|`. This means exactly one item from the set should be present.
- Three dots `...` denote that the item may be repeated.

2.1.4 The interpretation

Based on the description, the syntax given in Section 2.1.1 can be interpreted as follows:

1. First argument is the csv file name containing information about a course's registration
2. There are two optional flags (`--print` and `--find`). Each of them may by repeated.
   - The `--print` flag is always followed by *a name of a course* and *one of the words* `all`, `enrolled`, or `waitlist`.

- The `--find` flag is always followed by *a name of a student* and *a name of a course.*

The flag behaviors are explained in detail in Section 5.

3 Data

Each CSV contains information about a course. The first row contains information about the point distribution criteria, and the rest of the rows contain information about students pre-registering for the class.

3.1 First row

The first row of the registration csv describes how points should be calculated for a student. The data is in the following order:

```
class-name/enroll-capacity/waitlist-capacity, year-weight,
major-weight, minor-weight, BiCo-points, previous-lottery
```

As you already know, each class has a different specification of who is prioritized in the lottery. For example, seniors may be prioritized for certain courses (`year-weight`) or certain majors and minors have distinct levels of priorities (`major-weight` and `minor-weight`).

For simplicity, we will consider majors separate from class years and pretend that there are only 3 majors and 3 minors possible: `H`, `N`, and `S`.

The format of the columns will be as follows:

- `year-weight` is a 4 digit number, where first number represents the points for first-year students and last year represents the points for seniors.
- `major-weight` and `minor-weight` have a field-to-value mapping with a colon in between. See Section 3.3 below.
- `Bi-Co-points` (points awarded for Bi-Co students) and `previous-lottery` (points for previously being lotteried out of a course) are each a single integer.

Every file is guaranteed to have at least one row (the first row). Except for major and minor weights, every field is guaranteed not to be empty – but they may be 0. For example, the `waitlist-capacity` or `BiCo-points` maybe 0, but they will never be missing.

3.2 Rest of the rows

Each row represents a unique student (even if they share multiple attributes) and has the following information in order:

```
student-initials, class-year, majors,
minors, institution, previous-lottery
```

The student may have multiple majors or minors. If multiple values exist for these fields, they are separated with a slash (`/`).

3.3 Example

In the applicable columns, the point distribution is specified in the following manner: `<criterion>:points`, with criteria being separated by a slash (`/`). For example, consider the following course csv data below (only first 3 rows are shown):

```
"CS231/32/40", "4/4/1/0", "N:3/H:1", "N:1", "1", "2"
"CWL", "2", "", "", "HC", "No"
"BMH", "4", "N/S", "S", "BMC", "Yes"
```

The first row can be interpreted as follows. The course name is `CS231` with an enrollment capacity of `32` and a waitlist capacity of `40`. If a student is a first-year (`4/4/1/0`) or a sophomore (`4/4/1/0`), add 4 points. If the student is a junior (`4/4/1/0`), add 1 point. `N` majors get additional 3 points, and `H` majors get additional 1 point. `N` minors get additional 1 point. The students in the Bi-Co are given additional 1 point. Lastly, if a student has been previously lotteried out for a class, they would get additional 2 points.

The next two rows describe students with initials `CWL` and `BMH`. According to the criteria described in the first row, `CWL` would have 5 points (4 as a sophomore, 1 as a Bi-Co student) and `BMH` would have 6 points (1 as a Bi-Co student, 2 from being previously lotteried out, and 3 from being an `N` major).

4 Ordering and ties

The order of students in the lottery linked list should be as follows:

1. Students with higher scores come before students with lower scores.

2. If two nodes have the same score, then the node with a lexicographically earlier name (initials) comes before nodes with lexicographically later names. (e.g., SL comes before SLM, and BIB comes before BOB).

3. If two nodes have the same score and same lexicographic ordering of the name, then the nodes with earlier class years should come before the nodes with later class years.

Comes "before" means the student would be closer to the head than the student that comes "after".

Even if two students have the same amount of points, they should have a different position in the lottery (since each person has a unique number in the lottery in real life). For the basic (no extensions) lab, the position in the lottery should be determined by the order described above.

5 End Goal: Flags and Output

In the end, your `Main` class should output information based on the input. This section describes output behavior for the two flags, `--print` and `--find`, mentioned in Section 2.1.1. Dealing with ties will be discussed in Tasks instructions.

As always, run the appropriate `Verify` classes and get a passing status. Make changes to **your code** (**not** the `Verify` files) as necessary.

5.1 The `--find` flag

Recall that two arguments follow the `--find` flag: `<studentName>` and `<courseName>`. For each of the `--find` flag arguments, we want to output the `<studentName>`'s position in the lottery for `<courseName>`.

- If the student is fifth on the Waitlist, output should be in the following format: `<studentName>'s position in <courseName> lottery is W5.`

- If the student is third that is Enrolled in the class, output should be in the following format: `<studentName>'s position in <courseName> lottery is E3.`

- If the student is not enrolled or on the waitlist, the output should be in the following format: `<studentName> is not on the lottery for <courseName>.`

For the basic (no extensions) lab, you can ignore the `courseName` argument. Also, if there are multiple students with the same name (initials), then you should only print the **first** student in the lottery with the given name.

5.2 The `--print` flag

(You can ignore the arguments following `print` for the basic lab.) Recall that two arguments follow the `--print` flag: `<courseName>` and {`all | enrolled | waitlist`} (we'll call these options `modes`). For each of the `--print` flag items, we want to print the lottery results for the course `courseName`. Your output for the `--print` flag with the `all` argument should be in the following format:

```
<course-name> Lottery: <mode>
E1. <student-initials> <class-year> <points>
E2. <student-initials> <class-year> <points>
...
En. <student-initials> <class-year> <points>
W1. <student-initials> <class-year> <points>
W2. <student-initials> <class-year> <points>
...
Wm. <student-initials> <class-year> <points>
```

where `n` goes up to the enrollment (`E`) capacity and `m` goes up to the waitlist (`W`) capacity. If there are less students than the capacity, then `n` or `m` should not exceed the the capacity.

The output's header (first line) for the basic (no extensions) lab should always print `all` for `<mode>`, and give an output as above. The basic lab can also ignore the next paragraph of this document.

Note that the above format applies only if the `all` argument is included. If `enrolled` is the argument, then the list with numbers with prefix `E` should be printed out. If `waitlist` is the argument, then the list should have numbers with the prefix `W`.

The header should always be printed, even if the requested list is empty.

6 Example files

Some lottery csv data files are provided with the starter repository. There is also a folder named "example-output" which contain output samples for `lottery-csh231.csv`, `math224 lottery.csv`, and `socl182 lottery.csv`. You may find it helpful to look at the data and the output side-by-side.

Each output file contains examples of `find` and `print` flags for the **basic** version of the lab (without any extensions). These are also used for `Verify` files.

In the output file, the lines that start with `>>>` are the given command line arguments. The lines that follow are the output of a correct program.