COMP2123B Programming Technologies and Tools

Assignment 2

Deadline: 18 November 2018 (Sunday) 23:55

***** Please complete this question using Virtual Programming Lab (VPL) provided on Moodle. *****

Important note: You MUST use STL `vector` in your program. Your program will be checked after the submission deadline. Your score will be set to 0 if we find that you use traditional arrays to store pages and users in your implementation.

STL - Log Analyzer

Problem specification

Write a program, `logAnalyzer.cpp`, that reads and processes a web log data file from user input (`cin`). The program will print out the 5 most popular pages, and the 5 most active users and the corresponding pages they have visited.

The log file consists of records of three types, each record occupies exactly one line. Here is the format of these three types of record:

Page record: `PAGE` *`<page id> <page url>`*
User record: `USER` *`<user id>`*
Visiting record: `VISIT` *`<path id>`*

A **page** record represents a page on the web server. A **user** record represents a user that accesses the system. A **visiting** record represents a visit by the user indicated by the most recent user record. Here is a sample data file:

```
PAGE 1288 /library
PAGE 1282 /home
USER 20686
VISIT 1288
VISIT 1282
USER 20687
VISIT 1288
```

In this case, we have 2 pages (`/library` and `/home`) on the server. Two users have accessed the server, one (#20686) visiting 2 pages (`/library` and `/home`) and the other (#20687) visiting 1 page (`/library`).

You can assume the followings regarding the data file:

- The data file always consists of the three types of records only
- The page ids are unique across all PAGE records
- The user ids are unique across all USER records
- The page ids are unique across all VISIT records of a user
- VISIT records will only appear after the first USER record
- The page id in a VISIT record appears only after its corresponding PAGE record
- There will be at least 5 users and 5 pages

You are required to do the following in your program:

- Create a Page class, each Page object consists of the id, path and a counter to count the number times it is being visited

```
class Page {
    public:
        int id;
        string path;
        int counter;
        Page(int id, string path) {
            …
        };
        friend bool operator<(const Page & a,
                              const Page & b);
         // This function can facilitate sorting.
};
```

- Use STL vector to organize the Page objects

```
vector<Page> pages;
```

- Create a User class, each User object consists of the id and the pages the user visited

```
class User {
    public:
        int id;
        vector<string> visits;
        User(int id) {
            …
        };
        void add_visit(int page_id) {
            …
        };
        int size() const {
            return visits.size();
        };
        void print_visits() {
```

```
            …
        }
        friend bool operator<(const User & a,
                              const User & b);
        // This function can facilitate sorting.
};
```

- Use STL vector to organize the User objects

```
vector<User> users;
```

Sample Run

Assume we have `webdata1.txt` in the current directory (available on Moodle).
If we run the program like this: (assuming that we are working on Linux)

```
$ ./logAnalyzer < webdata1.txt
```

The program will print:

```
*** 5 most popular pages ***
36:/msdownload
32:/ie
17:/products
17:/search
13:/sitebuilder
*** 5 most active users ***
23:10068
- /activeplatform
- /activex
- /automap
- /frontpage
- /gallery
- /ie
- /intdev
- /isapi
- /java
- /msdownload
- /musicproducer
- /office
- /promo
- /sbnmember
- /search
- /sitebuilder
- /spain
- /vbasic
...
[snipped]
```

```
...
11:10019
- /athome
- /clipgallerylive
- /games
- /isapi
- /kb
- /logostore
- /msdownload
- /msoffice
- /ntserver
- /products
- /search
```

Note

- If two pages are equally popular, the pages are ordered lexicographically.
- If two users visit the same number of pages, the users are ordered by their id ascendingly.
- The list of pages a user visit must be printed in ascending order.