CS201 – Spring 2023 - Sabancı University

Homework #6: Bus Database Search

Due May 25, Thursday, 22:00

Brief Description

First let us start with a brief definition of the Database concept. A database is a collection of data records stored in a computer in a systematic way. In this way, a computer program can be used to answer queries about the data stored in the database by searching the records in a systematic fashion.

In this homework, you will write a program that will search bus routes and time databases to find departure times for given inputs. The database records that you will deal with are stored in two text files (see "Structure of Database (Input) Files" section for details). Your program will basically search and return bus times and destinations for a given departure point after a certain time. Resulting list will be displayed on the console as output (see "Database Search/Filter" section for details).

After successfully displaying the search results on the console, your program will ask parameters for another search until "EXIT" is entered as a termination command for departure location.

After a termination command, your program will close. Please check the "Sample Runs" below.

Your homework will be automatically graded using SUCourse, so it is very important to satisfy the exact same outputs given in the example test cases of SUCourse. Please submit your assignment by writing your main source (cpp) file content into the Answer field. You can utilize the **Check** button under the code editor at SUCourse to check whether your implementation is working in the expected way. After you check your solution code, you will see your grade with the example test cases used; however your homework will then be graded with **different** test cases.

To submit your homework, you must hit the **"Finish attempt…"** and **"Submit all and finish"** buttons. **Just a reminder of a character ↩ which refers to a newline in your expected output.**

<u>VERY IMPORTANT!</u>

Your programs will be compiled, executed and evaluated automatically; therefore, you should definitely follow the rules for prompts, inputs and outputs. See **<u>Sample Runs</u>** section for some examples.

<u>Order</u> of inputs and outputs must be in the abovementioned format.
<u>Prompts</u> before inputs and outputs must be **<u>exactly the same</u>** with examples.

Following these rules is crucial for grading, otherwise our software will not be able to process your outputs and you will lose some grades in the best scenario.

Input Check

There are three input checks in this homework:

- First you will ask for the name of the file that contains the routes database. If no file will be found, you should print out an error message and then ask for another file name for the routes database.
- The above rule applies for the times database as well.
- The last input check is on the format of the departure time. Time should be in format hh:mm, *hh* for 2-digit hour and *mm* for 2-digit minutes. Example: 03:05 not 3:5 or not 3:05. Additionally, the time needs to be in the valid range (between 00:00 and 23:59) such as 12:15, 21:03. Some examples of invalid time: 26:84, 10:60, 3a:0x, bb:dd. If this input is wrong, then your program should display an error message and ask for another departure time. (See Sample Runs for examples).
  Please NOTE that this time specific input check is only necessary for user entered input and not for the times database file. In the times database file, you may assume that all departure times are in the correct time format.

Structure of Database Files

As mentioned before, the database is stored in two different text files. One is only for route records, the other one is for route-time pair records. These two files will be input files to your program. (i.e. your program will read some data from both files)

Name of both input files will be entered by the user of your program. At the beginning of your program, users will be asked to enter an input file name for the route records. If the user enters a name for a file that does not exist, your program should display an appropriate message and ask for the name of a new input file name until the file is successfully opened. After the first input file is opened successfully, your program will ask for another input file name for the time records file. The same file opening control check will be performed for this file too. If a user enters a name for a file that does not exist, your program should display an appropriate message and ask for the name of a new input file name until the file is successfully opened.

Route Database File

The routes input file contains several information fields for each route. For a single route, there are three columns of information on the same line. Each line corresponds to the record of a route. The record structure for each route is described as follows:

```
routeID routeStart routeEnd
```

Record line starts with a ***routeID***. *routeID*'s are integer values and they are unique in each line. You will need to store this information to match the routes to times in the next file. It is followed by two strings: ***routeStart*** and ***routeEnd***.

The abovementioned record structure, which contains 3 different data items in one line, is just for one route. The input file contains information about several routes. Therefore, there are several such records in the input file. You may assume there is a record in each line of the input file, no line is empty. Also please note that the database file is not sorted (ordered).

Times Database File

The times database file contains time-route pairs written by that time. For a single time record, there are 2 pieces of information on the same line. The record structure for each route-time pair is described as follows:

```
routeID     departureTime
```

- ***routeID*** is a single integer value.
- ***departureTime*** is a string value. It is in the format 00:00. (ex: 03:02, 16:45 etc.)

We assume that **no** time-route pair appears **more than once** in the Times database file. For example, if there is a pair such as ***Route11 Time1,*** then there will **not** be another pair such as ***Route11 Time1***.

There is no specific order of records in both input files. That means you cannot assume that the records of the input files are ordered according to **routeID**, ***departureTime*** or any other data field.

The structure of the input files and the associated rules and assumptions explained above are fixed such that you cannot change them or make any other assumptions. Y**ou can assume that the input files will not be empty.** You may examine the sample input files provided in the homework google folder for this homework. Please note that we have multiple input files for multiple scenarios, therefore your code should also work with all these possible outcomes.

Important! There might be any number of white spaces (i.e. blanks) and/or tabs between and after each data item in both files. You should handle it.

Database Search/Filter

Database search will basically be searching the routes database file for all routes after a given departure time. First you need to specify a Departure Location for your search. After entering the departure location, (as long as it is not EXIT) you will be asked to enter a Departure Time to search for. You should only display the routes that are equal to or after this time. For example, if there are three bus routes from canakkale at times: 10:00, 12:00, 14:00 and the departure time is given as 12:00; then you should only print out the routes at 12:00 and 14:00 to the console.

Your results will contain **all** routes possible from the given departure location after (or equal to) the given departure time.

To do so, you must first search routes.txt and check each *routeStart* value to see if it is equal to the departure location given as input. If a match is found, you should store *routeID* and *routeEnd*. Then you should check the times database file to check if there is a corresponding bus route with that stored *routeID*. If a record is found, then you should check the time on that record with the user input departure time. If the route time is later than or equal to the departure time, then you should display that search result on a separate line to the console as specified below. The format of this line is as follows:

```
departureLocation routeEnd busTime
```

- ***departureLocation*** is the user's input,
- ***routeEnd*** is extracted from the routes database file,
- ***busTime*** is extracted from the times database file.

There is **only one** blank (white space) between each of these words. Please see the sample runs.

There will be any number of such lines, one for each route found, displayed on the console.

After successfully displaying one search result on console, your program will ask parameters for another search until "EXIT" word is entered as a termination command for departure location.

After a termination command, your program will close. You should check the “Sample Runs” below.

Some Hints

Maybe you have already realized that the simplest way of processing the input file is reading the data line by line and parsing each line. In order to perform such processing, the ideal mechanism is to use a combination of getline function and input string streams (istringstream). We have given/will give examples of such a processing in the lectures and recitations.

Note that string comparisons are **case sensitive**. That means *turkey* and *Turkey* are not equal to each other.

Please see sample runs for examples.

Use of Functions and Other Rules

Unlike the second homework, we will not specify any functions here. But you are expected to use functions to avoid code duplication and improve the modularity of your program. **If your main function or any user-defined function is too long and if you do everything in main or in another user-defined function, your grade may be lowered.**

AND PLEASE DO NOT WRITE EVERYTHING IN MAIN AND THEN TRY TO SPLIT THE TASK INTO SOME FUNCTIONS JUST TO HAVE SOME FUNCTIONS OTHER THAN MAIN. THIS IS TOTALLY AGAINST THE IDEA OF FUNCTIONAL DESIGN AND NOTHING BUT A DIRTY TRICK TO GET SOME POINTS. INSTEAD PLEASE DESIGN YOUR PROGRAM BY CONSIDERING NECESSARY FUNCTIONS AT THE BEGINNING.

IMPORTANT!

If your code does not compile, you will get **zero**. Please be careful about this and double check your code before submission.

No abrupt program termination please!

You may want to stop the execution of the program at a specific place in the program. Although there are ways of doing this in C++, it is not a good programming practice to abruptly stop the execution in the middle of the program. Therefore, your program flow should continue until the end of the main function and finish there.

Sample Run

Below, we provide a sample run of the program that you will develop. The italic and bold phrases are inputs taken from the user. You should follow the input order in these examples and the prompts your program will display must be **exactly the same** as in the following examples.

```
Please enter a filename for route database: r
cannot open routes database file
Please enter a filename for route database: routes
cannot open routes database file
Please enter a filename for route database: routes.txt
Please enter a filename for time database: time.txt
cannot open times database file
Please enter a filename for time database: time
cannot open times database file
Please enter a filename for time database: times.txt
Please enter departure location: Istanbul
Please enter start time of travel: 03:45
The search results are:
Istanbul Ankara 12:00
Istanbul Izmir 15:00
Istanbul Trabzon 11:00
Istanbul Trabzon 07:15
Please enter departure location: Adana
Please enter start time of travel: aa:15
Time is not in correct format
Please enter start time of travel: :!A.E
Time is not in correct format
Please enter start time of travel: 34:60
Time is not in correct format
Please enter start time of travel: 20:80
Time is not in correct format
Please enter start time of travel: 29:01
Time is not in correct format
Please enter start time of travel: 1:15
Time is not in correct format
Please enter start time of travel: 01:15
The search results are:
Adana Ankara 19:00
Adana Ankara 23:45
Adana Ankara 22:00
Please enter departure location: Izmir
Please enter start time of travel: 14:23
The search results are:
Izmir Istanbul 21:00
Izmir Istanbul 18:00
Izmir Istanbul 16:30
Please enter departure location: Trabzon
Please enter start time of travel: 23:00
The search results are:
Please enter departure location: Ankara
Please enter start time of travel: 13:22
The search results are:
Ankara Adana 14:45
Ankara Adana 18:30
Ankara Istanbul 17:00
Ankara Istanbul 23:00
Ankara Gaziantep 22:45
Ankara Gaziantep 20:30
Please enter departure location: gaziantep
Please enter start time of travel: 07:30
The search results are:
Please enter departure location: Antalya
Please enter start time of travel: 12:00
The search results are:
Please enter departure location: Eskisehir
Please enter start time of travel: 10:104
Time is not in correct format
Please enter start time of travel: 10:10
The search results are:
Eskisehir Ankara 17:30
Eskisehir Ankara 14:00
Please enter departure location: Izmir
Please enter start time of travel: 19:09
The search results are:
Izmir Istanbul 21:00
Please enter departure location: Gaziantep
Please enter start time of travel: 23:59
The search results are:
Please enter departure location: EXIT
```

General Rules and Guidelines about Homework

The following rules and guidelines will be applicable to all homework unless otherwise noted.

How to get help?

You may ask questions to TAs (Teaching Assistants) or LAs (Learning Assistants) of CS201. Office hours of TAs/LAs are at the SUCourse.

What and Where to Submit

You can prepare (or at least test) your program using MS Visual Studio 2019 C++ (Windows users) or using XCode (macOS users).

- Your code will be automatically graded using SUCourse. Therefore, it is essential that you ensure your output matches the exact same outputs given in the example test cases provided by SUCourse.
- After writing your code, use the "Check" button located under the code editor in SUCourse to see your grade based on the example test cases used. This grade will give you an idea of how well your code is performing.
- Note that the example test cases used for checking your code are not the same as the ones used for grading your homework. Your final grade will be based on different test cases. Therefore, it is important that you carefully follow the instructions and ensure that your code is working correctly to achieve the best possible grade on your homework assignment.
- To submit your homework, click on the "Finish attempt..." button and then the "Submit all and finish" button. If you wish to submit again before the due date, you can press the "Re-attempt quiz" button.
- Submit your work **through SUCourse only**! You will receive no credits if you submit by any other means (email, paper, etc.).

Grading, Review and Objections

Be careful about the automatic grading: Your programs will be graded using an automated system. Therefore, you should follow the guidelines on the input and output order. Moreover, It is important to use the exact same text as provided in the example test case outputs from SUCourse. Otherwise, the automated grading process will fail for your homework, and you may get a zero, or in the best scenario, you will lose points.

Grading:

- There is NO late submission. You need to submit your homework before the deadline. Please be careful that SUCourse time and your computer time may have 1-2 minute differences. You need to take this time difference into consideration.
- Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.
- If your code does not work because of a syntax error, then we cannot grade it; and thus, your grade will be 0.
- Please submit your **own** work only. It is really easy to find "similar" programs!
- Plagiarism will not be tolerated. Please check our plagiarism policy given in the Syllabus.

Plagiarism will not be tolerated!

Grade announcements: Grades will be posted in SUCourse, and you will get an Announcement at the same time. You will find the grading policy and test cases in that announcement.

Grade objections: It is your right to object to your grade if you think there is a problem, but before making an objection please try the steps below and if you still think there is a problem, contact the TA that graded your homework from the email address provided in the comment section of your announced homework grade or attend the specified objection hour in your grade announcement.

- Check the comment section in the homework tab to see the problem with your homework.

- Check the test cases in the announcement and try them with your code.
- Compare your results with the given results in the announcement.

Good Luck!
CS201 Instructors