VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Assignment 1 - Phase 2

URBAN BUS SYSTEM

HO CHI MINH CITY, 11/2020

ASSIGNMENT 1 SPECIFICATION (Phase 2)

Version 1.0

1 Outcome

After successfully completing this assignment, students will be able to:

- Apply a known data structure to solve problems in real life.
- Analyse the complexity of the implemented algorithm based on empirical study.

2 Introduction

In big cities, public transport is an important object of the city's overall infrastructure. Public transport is one of the methods of transportation for modern people in big cities, especially buses.

The bus transport system has many routes. Each route will be assigned with a fixed code, originating stop (first stop), terminus (last stop) and travel time to travel across the route. Everyday, each route will have many buses going from the first to the last stop. To differentiate between buses on the same route, people use license plate of vehicles and departure time .

3 Tasks

Students are requested to build a program in C++ to store information about these bus routes, assuming that the maximum number of trips every day is equal, denoted as **N**. All of the data-processing commands on this program have the complexity of $O(\mathbf{N})$.

Students can re-implement another suitable data structure, but are encouraged to use the implemented **FragmentLinkedList** structure in Phase 1 for storage in this program.

4 Data-processing commands

4.1 Instruction

Each data-processing command is a sequence of characters. Each command begins with a keyword (the bold word in description) and is followed by parameters (the word in < and > in the description). There is exactly a space between the command and the parameters. There should be no spaces before the command keyword and no spaces after the last parameter. Some last parameters can be optional (with or without the command), these parameters will be placed in [and] in the description. When a command does not provide the correct number of parameters or does not have the correct type of the parameter as described or has spaces that are not as described, the command will not be executed and return string "-1". Otherwise, the command will be executed and a string will be returned according to the description in the section 4.2. (if the returned result is described as a number, convert it to a string for returning)

The meaning of the acronym and the data type of the parameters are described as follows:

- CODE: a sequence of characters representing the route code.
- LP: a sequence of characters that stores the value of the license plate of the bus.
- CASE: a binary value, if **true**, the bus goes from the start to the end (outward); otherwise, the bus goes in the opposite direction (return).
- TIME: an integer (representing time according to ISO standard) representing time.
- N: maximum number of trips for all routes.

In the commands which have two optional parameters of time <TIME_A> and <TIME_B> (denoted as [<TIME_A> [<TIME_B>]]), the program will execute the command with other parameters and

- <TIME_A>≤ <TIME>≤ <TIME_B> if there are both <TIME_A> and <TIME_B> in the command.
- <TIME> = <TIME_A> if there is only <TIME_A> in the command.
- every <TIME> if there is no parameter of <TIME_A> and <TIME_B>.

Each command should be processed with a time complexity not exceeding the specified complexity in the column "Complexity". After processing all the commands and returning the corresponding values for each command, the program must make sure to delete all dynamically allocated data objects, leave no trash in memory before terminating the program.

4.2 Commands list

Request	Complexity	Description
SQ <N>	O(**1**)	Set the maximum number of trips for all routes. This command is executed only one time in the program and before all other commands. If the command is executed successfully as described, return 1, otherwise return -1.
INS <CODE> <LP> [<CASE>] <TIME_A> <TIME_B>	O(**N**)	Insert a bus trip with license plate <LP> of the route with code <CODE> with departure time <TIME_A>, arrive at the last stop at <TIME_B>. The command is executed successfully when there is no bus trip on the same route with same departure time. In case the license plate has existed, the departure time <TIME_A> has to be later than the previous arrival time. When the command is executed successfully, return the number of saved license plates of the route, otherwise return -1. Assume that a license plate can only make trips on a fixed route.
DEL <CODE> [<TIME_A> [<TIME_B>]]	O(**N**)	Delete the saved trips of the route <CODE> which have the departure time between <TIME_A> and <TIME_B>. The command returns the number of trips deleted from the data set.
CS <CODE> <TIME> [<CASE>]	O(**N**)	Return the number of trips that have started but have not yet arrived at the bus stop of the route with code <CODE> at the considered time <TIME>. When we have value <CASE>, we need to return the correct number of trips corresponding to outward trips or return trips, otherwise we will return all the trips which are satisfied in both directions.
CE <CODE> <TIME> [<CASE>]	O(**N**)	Return the number of trips that have ended of the route with code <CODE> at the considered time <TIME>. When we have value <CASE>, we need to return the correct number of trips corresponding to outward trips or return trips, otherwise we will return all the trips which are satisfied in both directions.

GS <CODE> <TIME> [<CASE>]	O(**N**)	Return the license plate of the bus trip of the route with code <CODE> which has started at the time closest to the considered time <TIME>. When we have value <CASE>, we need to return the correct trip corresponding to outward trips or return trips, otherwise, we will consider both directions. In case there is no trip that is satisfied, return -1. In case considering both directions, if there are 2 satisfied trips, return the outward trip.
GE <CODE> <TIME> [<CASE]	O(**N**)	Return the license plate of the bus trip of the route with code <CODE> which has ended at the time closest to the considered time <TIME>. When we have value <CASE>, we need to return the correct trip corresponding to outward trips or return trips, otherwise, we will consider both directions. In case there is no trip that is satisfied, return -1. In case considering both directions, if there are 2 satisfied trips, return the outward trip.

5 Regulation and Submission

Students will be provided with an initialization code framework on the general course site. Students code directly on the BKeL system and the student's final submission will be used for grading.

All of the students' questions about this assignment will be answered on the course site forum. DO NOT SEND ANY EMAIL FOR QUESTIONING to the teachers or teaching assistants under any circumstances.

——————THE END——————