Assignment 4: Flight-Ticket Management System

Data Structures (CS-UH 1050) — Spring 2022

1 Code of Conduct

All assignments are graded, meaning we expect you to adhere to the academic integrity standards of NYU Abu Dhabi. To avoid any confusion regarding this, we will briefly state what is and isn't allowed when working on an assignment.
Any documents and program code that you submit must be fully written by yourself. You can discuss your work with fellow students, as long as **these discussions are restricted to general solution techniques, without sharing the overall or specific algorithms.** Put differently, these discussions should not be about concrete code you are writing, nor about specific set of steps or results you wish to submit. When discussing an assignment with others, this should never lead to you possessing the complete or partial solution of others, regardless of whether the solution is in paper or digital form, and independent of who made the solution, meaning you are also not allowed to possess solutions by someone from a different year or course, by someone from another university, or code from the Internet, etc. This also implies that **there is never a valid reason to share your code with fellow students**, and that there is no valid reason to publish your code online in any form. Every student is responsible for the work they submit. If there is any doubt during the grading about whether a student created the assignment themselves (e.g., if the solution matches that of others), **the suspected violations will be reported to the academic administration according to the policies of NYU Abu Dhabi** (see https://students.nyuad.nyu.edu/campus-life/community-standards/policies/academic-integrity/) under the integrity review process.

2 Introduction

In this assignment, you will develop a Flight-Ticket Management application that can help you search for a ticket that suits you. The identification of a flight-ticket will be based on the key produced by the combination of the company name and flight number.

Information of flight-tickets is to be inserted/read from either the interface of your application or from an input text file. The application should support deletion of a ticket record in addition to other management features that are highlighted in the following section.

Each record is provided in ASCII format and spans at most one line of text. The user interface should allow a user to navigate and perform all the tasks/features listed in the following section.

You are supposed to create a program utilizing the object-oriented programming (OOP) principles and appropriate data structures discussed in class. You are required to use **Hash Table** as the primary data structure to implement the application. Use **separate chaining** to handle collisions.

You should implement all the data structures manually, STL based containers are not allowed to be used here.

3 Implementation

CSV File Format:

The flight-ticket record consists of the following fields:

companyName, flightNumber, country_of_origin, country_of_destination, stopover, price, time_of_departure, time_of_arrival, date

Ex.: `Emirates Sky Cargo,1,Lebanon,USA,3,800,11:50,14:15,1-Apr`

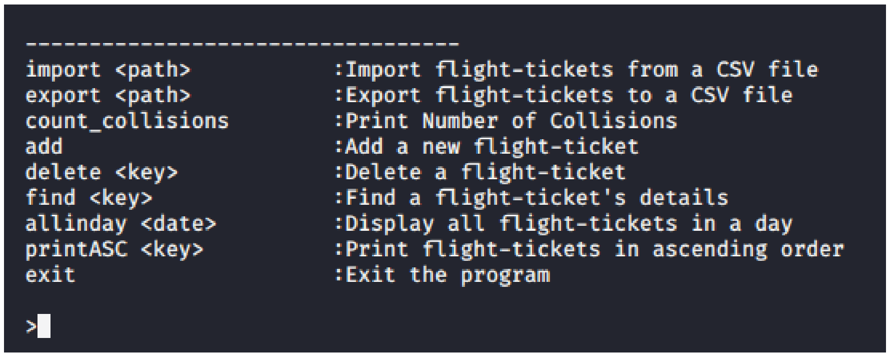
Examples of flights records:

```
Emirates Sky Cargo,1,Lebanon,USA,3,800,11:50,14:15,1-Apr
Aeromexico,70,Bahrain,Luxembourg,1,1328,0:04,3:16,16-Dec
Gol Transportes Aéreos,103,Sweden,Lebanon,3,1387,12:58,12:26,26-Aug
Boliviana de Aviación,43,Belarus,Cyprus,0,1157,23:02,0:07,9-Aug
Aeromexico,52,Mongolia,United States,3,685,3:34,21:27,24-Feb
Conviasa,75,Canada,Hungary,3,1411,1:02,18:24,21-Feb
```

User Interface:

Through its prompt, the program is expected to accept several commands that we outline next; these commands accept as operand a string that provides a flight-ticket’s data, or the combination of companyName and flightNumber used as the key to look up records.

The application interacts with the user using a command line (terminal based) interface as shown below:



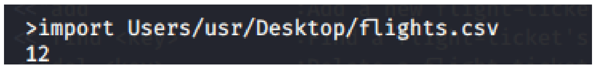
```
----------------------------------
import <path>           :Import flight-tickets from a CSV file
export <path>           :Export flight-tickets to a CSV file
count_collisions        :Print Number of Collisions
add                     :Add a new flight-ticket
delete <key>            :Delete a flight-ticket
find <key>              :Find a flight-ticket's details
allinday <date>         :Display all flight-tickets in a day
printASC <key>          :Print flight-tickets in ascending order
exit                    :Exit the program

>
```

The application, after performing the user's desired operation, presents the menu again for the next operation to be performed.

NOTE: Make sure to handle the cases of the input keys provided by the user, for example you can use the `tolower()` function. Moreover, the below screenshots are just examples for showing you samples of inputs and outputs, so you may have different results when executing your own code.

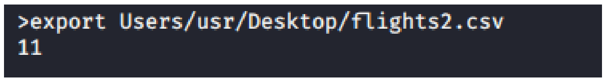
Features to Implement:

1. **import <path>**
   - Imports a CSV file
   - Invokes a function with the following header:
     `int importCSV (string path)`
   - This function takes the path as an argument and loads all the flight-tickets from the CSV file into the Hash Table. The function either returns the number of flight-tickets read from the file as integer or returns -1, if a bad path is provided.



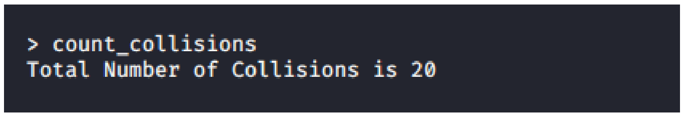
```
>import Users/usr/Desktop/flights.csv
12
```

2. **export <path>**
   - Exports a CSV file
   - Invokes a function with the following header:
     `int exportCSV (string path)`
   - This function takes the path as an argument and writes all the flight-tickets from the Hash Table to a CSV file. The function either returns the number of flight-tickets written to the file as integer or returns -1, if file cannot be created



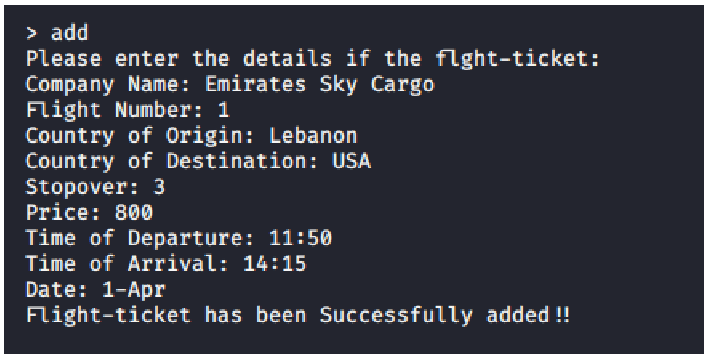
```
>export Users/usr/Desktop/flights2.csv
11
```

3. **count_collisions**
   - Returns the number of collisions caused by a hash function.
   - Invokes a function with the following header:
     `int count_collisions ()`
   - This function doesn't accept any parameters. It should operate based on the existing hash table. This method returns the number of collisions in the hash table.

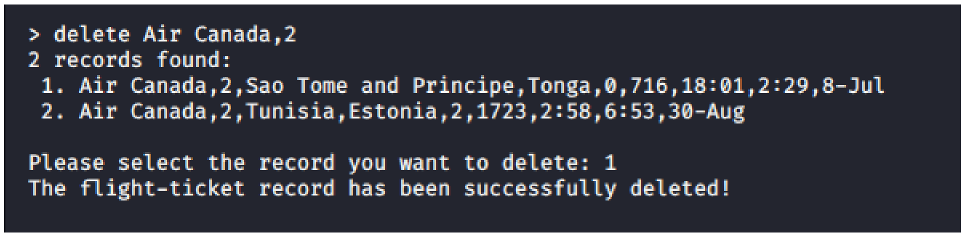


```
> count_collisions
Total Number of Collisions is 20
```

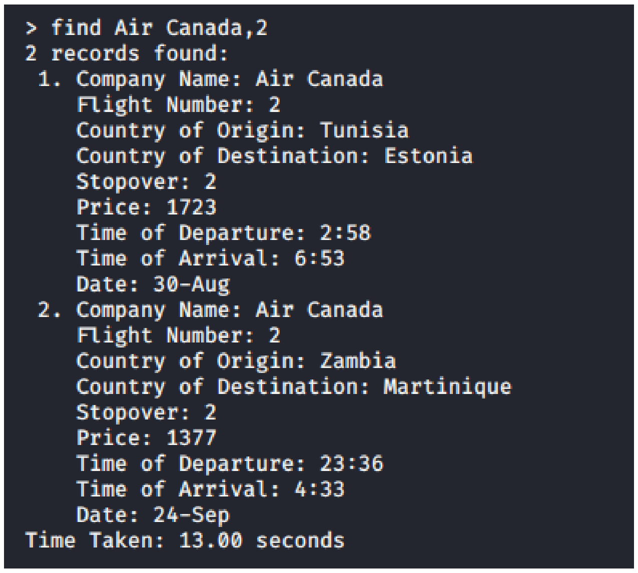
4. **add**
   - Adds a new flight record. If the element already exists, no entry will be added, but a warning message will be issued stating that the record already exists the system.
   - Invokes a function with the following header:
     `int add (FlightInfo *data)`
   - This function takes a pointer to a FlightInfo object as an argument. The function returns 1 if the record has been successfully added or -1 otherwise.


```
> add
Please enter the details if the flght-ticket:
Company Name: Emirates Sky Cargo
Flight Number: 1
Country of Origin: Lebanon
Country of Destination: USA
Stopover: 3
Price: 800
Time of Departure: 11:50
Time of Arrival: 14:15
Date: 1-Apr
Flight-ticket has been Successfully added !!
```

5. **delete <companyName,flightNumber>**
   - Deletes a flight-ticket record based on the entered key <companyName,flightNumber>. If no such entry exists, a warning message to the standard error is printed out.
   - Invokes a function with the following header:
     void removeRecord (string companyName, int flightNumber)
   - This function takes a key (i.e., companyName and flightNumber) as argument. It prints confirmation message if the record has been successfully added or failure message otherwise.

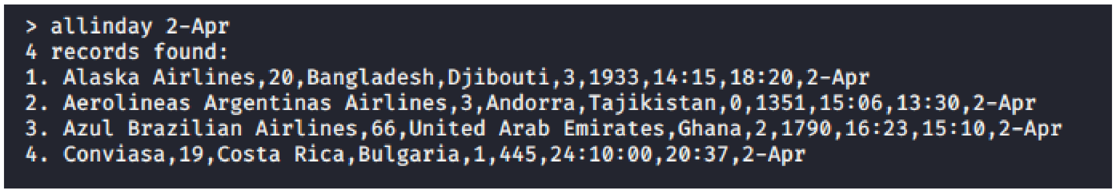

```
> delete Air Canada,2
2 records found:
 1. Air Canada,2,Sao Tome and Principe,Tonga,0,716,18:01,2:29,8-Jul
 2. Air Canada,2,Tunisia,Estonia,2,1723,2:58,6:53,30-Aug

Please select the record you want to delete: 1
The flight-ticket record has been successfully deleted!
```

6. **find <companyName,flightNumber>**
   - Searches for a flight-ticket record of the entered key <companyName,flightNumber>.
   - Invokes a function with the following header:
     void find (string companyName, int flightNumber)
   - This function takes a key (i.e., companyName and flightNumber) as an argument and displays the entire records that match the entered key or displays "*Not Found!*" message otherwise. It also prints out the actual time taken by the find execution.



```
> find Air Canada,2
2 records found:
 1. Company Name: Air Canada
    Flight Number: 2
    Country of Origin: Tunisia
    Country of Destination: Estonia
    Stopover: 2
    Price: 1723
    Time of Departure: 2:58
    Time of Arrival: 6:53
    Date: 30-Aug
 2. Company Name: Air Canada
    Flight Number: 2
    Country of Origin: Zambia
    Country of Destination: Martinique
    Stopover: 2
    Price: 1377
    Time of Departure: 23:36
    Time of Arrival: 4:33
    Date: 24-Sep
Time Taken: 13.00 seconds
```

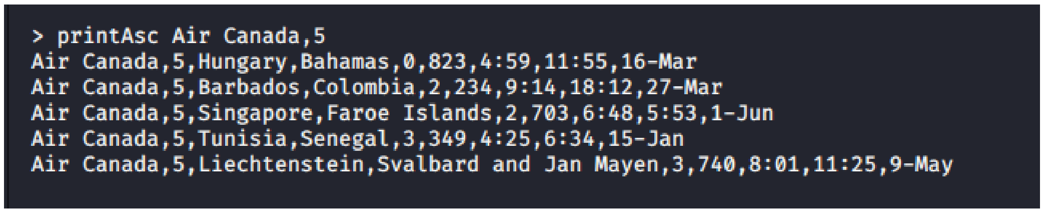
7. **allinday** **<date>**
   - Searches for all the Flight-tickets for a specified date.
   - Invokes a function with the following header:
     void allinday (string date)
   - This function takes a specific date as an argument and prints out all the flight-tickets available for that date or display "*Not Found!*" message otherwise.



```
> allinday 2-Apr
4 records found:
1. Alaska Airlines,20,Bangladesh,Djibouti,3,1933,14:15,18:20,2-Apr
2. Aerolineas Argentinas Airlines,3,Andorra,Tajikistan,0,1351,15:06,13:30,2-Apr
3. Azul Brazilian Airlines,66,United Arab Emirates,Ghana,2,1790,16:23,15:10,2-Apr
4. Conviasa,19,Costa Rica,Bulgaria,1,445,24:10:00,20:37,2-Apr
```

8. **printASC** **<companyName,flightNumber>**

   - Displays in an ascending order (based on the flight destination country) all flight-tickets having the same key entered by the user <companyName,flightNumber>.
   - Invokes a function with the following header:
     void printASC (string companyName, int flightNumber)



```
> printAsc Air Canada,5
Air Canada,5,Hungary,Bahamas,0,823,4:59,11:55,16-Mar
Air Canada,5,Barbados,Colombia,2,234,9:14,18:12,27-Mar
Air Canada,5,Singapore,Faroe Islands,2,703,6:48,5:53,1-Jun
Air Canada,5,Tunisia,Senegal,3,349,4:25,6:34,15-Jan
Air Canada,5,Liechtenstein,Svalbard and Jan Mayen,3,740,8:01,11:25,9-May
```

9. **exit**
   - Exits the program

3 Experiments and Report

- **You need to implement three different hash functions** and evaluate the performance of each hash function in terms of the number of collisions when applied on the same dataset files (use the supplementary files).
- The hash function causing the least number of collisions, on average when applied to the 3 dataset files, **should be set as the default hash function to use by your system**.
- You must document the description and design of each hash function along with their evaluation results in **a pdf report**.

4 Grading

Description	**Score (/20)**
Quality in Code Organization & Modularity - Defining ALL the needed classes correctly, with their sets of attributes and methods, and creating objects from each class properly - Dividing the programming properly in header and cpp files - Loading and reading the input file correctly - Updating the files properly upon exit - Implement the data structures yourself; don't use STLs	5
Correct design and implementation of the required methods: import (1 point), export (1 point), add (1 point), delete (1 point), find (1 point), count_collisions (1 point), allinday (1 point), printASC (1 point)	8
Clear and complete report (in pdf format) of the experimental results exploring the performance of the three hash functions when applied to the same 3 supplementary datasets in terms of the number of collisions per dataset and then the average in overall	3
Documentation (comprehensive comments on every code block in the program)	1
Submission should be <u>a single zip file</u> that includes *.cpp, *.h and Makefile files	1
Correct user interface implementation, compliance to the starter code and the example illustrations of the commands	2

You should solve and <u>**work individually**</u> on this assignment. <u>**The final hard deadline of this assignment is May 8**</u>. <u>**No late submissions will be accepted.**</u>

You should compress your **C++ source files** (*.cpp, *.h, makefile) into <u>**a single zip file**</u> before uploading it directly to NYU Brightspace. <u>**NO SUBMISSIONS OR RESUBMISSIONS VIA EMAIL WILL BE ACCEPTED.**</u> Note that your program should be implemented in C++ and **must be runnable on the Linux, Unix or macOS operating system**.