COMP2521: Assignment 1

ADTs: FlightDb Using a *Generic* AVL Tree

This specification may change! So please check the following change log frequently.

Change log:

- [11/03 12:50pm] Fixed style inconsistencies
- [12/03 11:20pm] Fixed inconsistent function names between `Record.c` and `Record.h`
- [13/03 07:40pm] Fixed `n->height` not being initialised in `newNode` in `AVLTree.c`
- [18/03 04:40pm] Added comments to `TreeNext` and `DbFindNextFlight` to clarify what should be returned if there is no appropriate record
- [18/03 04:50pm] Added sample tests
- [26/03 03:00pm] Added clarifications
- [30/03 12:55am] Added instructions on how to Submit this assignment (goto: Submission)

Objectives

- to give you experience in implementing ADTs in C
- to give you experience with higher order functions and generic ADTs
- to give you further practice with C and data structures

Admin

- **Marks:** 15 marks
- **Deadline:** 5pm ~~Friday~~ Saturday of Week 07
- **Late Penalty:** 1.5 marks per day off the ceiling. For example, if you submit any time between 1 and 2 days late, the maximum mark you can obtain is 12/15. The latest you can submit is 5pm Monday of Week 08. After this date, you will receive zero for the assignment.
- **Individual Assignment:** This assignment is completed individually.

Aim

In this assignment, your task is to implement functions for the AVLTree and FlightDb ADTs, in the files `AVLTree.c` and `FlightDb.c`. You will have to write wrapper code to test your implementations, and make sure to test a variety of conditions.

Please note that, you need to submit only the C code implementing the required functions in the files `AVLTree.c` and `FlightDb.c`.

Week 04 Lab

You should first complete the Week 04 lab exercise. This assignment is based on many concepts discussed in the Week 04 lab, like comparison functions, function pointers and their usage in implementing Tree and StudentDb ADTs. In this assignment, you need to use similar setup, however here you need to implement the AVLTree and FlightDb ADTs.

Clarifications

- You can assume that flights with the same flight number will **always** have the same departure and arrival airports. You do not need to check this in your code.
- You are always allowed to use helper functions. Since you are not allowed to modify anything above "the line" in `AVLTree.c`, please put the prototypes for your helper functions under this line.
- You may not use `string.h` in `AVLTree.c`. Doing this is a sign that your code is incorrect.
- For `DbFindBetweenTimes`, the times are *inclusive*.
- For `DbFindBetweenTimes`, you will never be passed two times where the first time is later than the second.
- For `DbFindNextFlight`, if there are multiple flights that occur at the same time that could be selected from, then you must take the one with the smallest flight number.

Task 1: AVLTree ADT

The AVLTree ADT implements an AVL tree of records with a custom comparison function.

You need to implement the following **three functions** in the file `AVLTree.c`.

```
/**
 * Inserts the given record into the AVL tree.
 * You  must use the proper AVL insertion algorithm, as discussed in the
 * lectures.
 * The time complexity of this function must be O(log n).
 * Returns true if the record was inserted  successfully,  or  false  if
 * there was already a record that compares equal to the given record in
 * the tree (according to the comparison function).
 */
bool   TreeInsert(Tree t, Record rec);

/**
 * Searches  for  all  records  between the two given records, inclusive
 * (according to the comparison function) and returns the records  in  a
 * list  in  order.  Returns an empty list if there are no such records.
 * Assumes that `lower` is less than `upper`.
 * The time complexity of this function must be O(log n + m), where m is
 * the length of the returned list.
 */
List   TreeSearchBetween(Tree t, Record lower, Record upper);

/**
 * Returns the smallest record greater than or equal to the given record
 * r (according to the comparison function), or NULL if there is no such
 * record.
 * The time complexity of this function must be O(log n).
 */
Record TreeNext(Tree t, Record r);
```

Please make sure you follow the following instructions:

- You must use the the proper AVL insertion algorithm, as discussed in the lectures, for inserting a record.
- Do **not modify** `AVLTree.h`. You will only submit `AVLTree.c` for this task.
- In `AVLTree.c`, do **not modify** the structures defined for 'node' and 'tree'.
- You must **not modify** the provided functions in AVLTree.c (from line 1 to 100).

Task 2: FlightDb ADT

FlightDb ADT stores flight details (see `Record.h` and `Record.c` for more information), and offers efficient operations for the searching tasks defined in the interface `FlightDb.h`. Your task is to implement the interface `FlightDb.h` in the file `FlightDb.c`. You MUST use the AVLTree ADT (from Task 1) in your implementation. You MUST only create one record per flight, like you did in Week-04 lab.

In this assignment, our focus is on learning how to implement an efficient database ADT using an AVL tree ADT. So we assume the following regarding the flight details:

- flight schedule is the same every week
- departureDay is between 0 and 6. 0 for Monday, 1 for Tuesday, ... , 6 for Sunday
- departureHour is between 0 and 23 (inclusive)
- departureMinute is between 0 to 59 (inclusive)
- durationMinutes indicates the duration of the flight in minutes

Please do NOT modify the header file `FlightDb.h`, you will only submit `FlightDb.c` for this task. We will test your implementation using our test sets, so make sure you do NOT modify function headers (signatures) in the header file `FlightDb.h`.

You need to implement the following functions in `FlightDb.c`.

```
/**
 * Creates a new flight DB.
 * You MUST use the AVLTree ADT (from Task 1) in your implementation.
 */
FlightDb DbNew(void);

/**
 * Frees all memory allocated to the given flight DB
 */
void     DbFree(FlightDb db);

/**
 * Inserts  a  flight  record  into the given DB if there is not already
 * record with the same flight number, departure airport, day, hour  and
 * minute.
 * If  inserted successfully, this function takes ownership of the given
 * record (so the caller should not modify or free it).
 * Returns true if the record was successfully inserted,  and  false  if
 * the  DB  already  contained  a  record  with  the same flight number,
 * departure airport, day, hour and minute.
 * The time complexity of this function must be O(log n).
 * You MUST use the AVLTree ADT (from Task 1) in your implementation.
 */
bool     DbInsertRecord(FlightDb db, Record r);

/**
 * Searches  for  all  records with the given flight number, and returns
 * them all in a list in increasing order of  (day, hour, min).  Returns
 * an empty list if there are no such records.
 * The  records  in the returned list should not be freed, but it is the
 * caller's responsibility to free the list itself.
 * The time complexity of this function must be O(log n + m), where m is
 * the length of the returned list.
 * You MUST use the AVLTree ADT (from Task 1) in your implementation.
 */
```

```
List     DbFindByFlightNumber(FlightDb db, char *flightNumber);

/**
 * Searches  for all records with the given departure airport and day of
 * week (0 to 6), and returns them all in a list in increasing order  of
 * (hour, min, flight number).
 * Returns an empty list if there are no such records.
 * The  records  in the returned list should not be freed, but it is the
 * caller's responsibility to free the list itself.
 * The time complexity of this function must be O(log n + m), where m is
 * the length of the returned list.
 * You MUST use the AVLTree ADT (from Task 1) in your implementation.
 */
List     DbFindByDepartureAirportDay(FlightDb db, char *departureAirport,
                                     int day);

/**
 * Searches  for  all  records  between  (day1, hour1, min1)  and (day2,
 * hour2, min2), and returns them all in a list in increasing  order  of
 * (day, hour, min, flight number).
 * Returns an empty list if there are no such records.
 * The  records  in the returned list should not be freed, but it is the
 * caller's responsibility to free the list itself.
 * The time complexity of this function must be O(log n + m), where m is
 * the length of the returned list.
 * You MUST use the AVLTree ADT (from Task 1) in your implementation.
 */
List     DbFindBetweenTimes(FlightDb db,
                            int day1, int hour1, int min1,
                            int day2, int hour2, int min2);

/**
 * Searches  for  and  returns  the  earliest next flight from the given
 * departure airport, on or after the given (day, hour, min), or NULL if
 * there is no such flight.
 * The returned record must not be freed or modified.
 * The time complexity of this function must be O(log n).
 * You MUST use the AVLTree ADT (from Task 1) in your implementation.
 */
Record   DbFindNextFlight(FlightDb db, char *departureAirport,
                          int day, int hour, int min);
```

Setting Up

Create a directory for this assignment, change into it, and run the following command:

```
unzip /web/cs2521/21T1/assigns/ass1/downloads/ass1.zip
```

If you're working at home, download ass1.zip by clicking on the above link and then run the unzip command on the downloaded file.

If you've done the above correctly, you should now have the following files:

`List.c`
: implementation of the `List` ADT (complete)

`List.h`
: interface to the `List` ADT

`Record.c`
: implementation of the `Record` ADT (complete)

`Record.h`
: interface to the `Record` ADT

`FlightDb.c`
: implementation of the `FlightDb` ADT (incomplete)

`FlightDb.h`
: interface to the `FlightDb` ADT

`AVLTree.c`
: implementation of the `AVLTree` ADT (incomplete)

`AVLTree.h`
: interface to the `Tree` ADT

Testing

You can download the sample tests by running the following command from your assignment directory.

Important: The test files provide new versions of `Record.c` and `Makefile`. If you made changes to any of these files, you should create backups of these first.

```
unzip /web/cs2521/21T1/assigns/ass1/downloads/testing.zip
```

If you're working at home, download testing.zip by clicking on the above link and then run the unzip command on the downloaded file.

You should now have the following *new* files:

`testAss1.c`
: a program that contains some sample tests

`output/`
: a directory containing expected output for the tests in `testAss1.c`

To test your implementation, run the `make` command to compile your code and then run `./testAss1` *`test-number`* (for example, `./testAss1 1`). Then compare the output to the expected output in the `output/` directory. The expected output for test *n* is in `output/`*`n`*`.exp`.

Please note that the sample tests are by no means comprehensive - you should do your own testing by adding tests to this file. To avoid modifying the existing tests, you should create functions containing your tests, and extend the switch statement in the `main` function to call these functions. Of course, there won't be any expected output for these tests so you will need to inspect the output yourself.

The sample tests also do not directly test the AVLTree ADT implementation. However, it is possible to test your AVLTree by creating a tree (you'll need to create a comparison function first), inserting records and then calling the various AVLTree ADT functions.

Assessment Criteria

Your program will be tested thoroughly and objectively. This assignment will be marked out of 15 where 12 marks are for correctness and 3 marks are for your solution's complexity, style and comments. Task 1 is worth 5 marks and Task 2 is worth 10 marks.

Submission

You need to submit two files: **`AVLTree.c`** and **`FlightDb.c`**.

To Submit this assignment, go to the **Submission page** and click the tab named "**Make Submission**".

You can also submit at the command-line, by running

```
give cs2521 ass1 AVLTree.c FlightDb.c
```

Plagiarism

This is an individual assignment. Each student will have to develop their own solution without help from other people. You are not permitted to exchange code or pseudocode. If you have questions about the assignment, ask your tutor. All work submitted for assessment must be entirely your own work. We regard unacknowledged copying of material, in whole or part, as an extremely serious offence. For further information, read the Student Conduct.
