Computer Science II — CSci 1200
Homework 6 — Rensselaer Airlines
Due: November 2, 2004

This homework is worth 100 points. The solution is due by 11:59:59pm on Tuesday, November 2. See the handout on homework and programming guidelines for style, submission instructions, and grading criteria. Remember to zip your files together prior to submission and then submit only the zip file. If you use some of the code provided (see below), be sure to include these files as well!

Problem

Your problem is to represent the travel schedule for passengers and flights on an airline in a single day. Passengers can take as many flights in one 24 hour period as they can fit, but these flights must be connected in a chain. In other words, except for the first flight, each flight must depart from the arrival city of the previous flight. Moreover, to keep things simple, the flights must be scheduled in order, from first to last. Finally, there must be at least 30 minutes between the passenger's arrival in a city and his/her departure on a subsequent flight from the same city.

Several other simplifying assumptions will make this problem solvable in the time allotted:

- Once a passenger is scheduled for a flight, s/he can not be removed from the flight.
- Time is represented in hours and minutes, with hours in the range 0..23 and minutes in the range 0..59. There is no need to worry about time zones.
- All flights originate and complete within a single day's 24 hours.
- For each flight there is a single price for each passenger.

Input of Flights

All input will come from `cin`. The first input will be the flights, with one flight per line. Each flight will be specified by an id number (please represent this as an integer), a departure city, a departure time, an arrival city, an arrival time, a cost, and the maximum number of passengers who can take the flight. For example,

```
143 Chicago 11 45 Atlanta 14 05 297.99 155
```

means that Flight 143 leaves Chicago at 11:45 and arrives in Atlanta at 14:05. The flight costs $297.99 and has a capacity of 155 passengers.

Continue reading flight information until a flight id of 0 is provided. The remainder of the input line for this id will contain only whitespace characters so don't try to read anymore information about this flight.

You may assume:

- Each flight will have a different id.
- Each flight is between exactly two cities, meaning that a flight does not continue on to subsequent arrival cities.
- City names are single strings (for example, `San Francisco` would be input as `SanFrancisco`), and there will be no misspellings and no changes in the capitalization of a city name.
- All input about the flights will be correct.

Scheduling and Queries

Once your program has input the flight information, it should proceed to handling input queries. The following queries will be specified, one per line of input. In all cases, the input format will be correct.

0 flight-id first-name last-name — Add the passenger with the given first and last name to the flight. If the flight does not exist, please output the following error message on a separate line:

```
No flight flight-id
```

where `flight-id` is the id provided on the input line. If the flight is full, please output the following error message:

```
Flight flight-id is full
```

If adding this flight is inconsistent with the passenger's current travel schedule, according to the criteria specifed above, then output the error message

```
Conflicts with schedule of first-name last-name
```

(Note that if the passenger is already scheduled on this flight, there will be a schedule conflict according to the rules outlined above. Hence, you do not have to explicitly check if the passenger is already scheduled on the flight.) Otherwise, add the passenger to the flight and output the line

```
first-name last-name added to Flight flight-id
```

For each of the these output lines, `first-name`, `last-name` and `flight-id` should be the appropriate strings from the input. You may assume each name is a single string.

1 city1 city2 — List all flights from `city1` to `city2`, ordered by flight id string. (Ordering by id makes the problem easier.) Start by outputing the line

```
Flights between city1 and city2
```

Output one flight per line. For each flight, output the flight-id, the departure time, the arrival time, the cost, and the number of seats remaining on the flight. If there are no flights between cities, please output the line

```
none
```

2 first-name last-name — Output the travel schedule for the given passenger. If the passenger has not scheduled any flights, output the message

```
first-name last-name has no travel schedule
```

Otherwise output the line

```
Travel schedule for first-name last-name
```

Then, for each flight, output the flight id, the departure city, the departure time, the arrival city and the arrival time, all on one line of output. After output of the flight information, output the total amount of money the passenger owes.

3 flight-id — If the flight id does not exist, output

```
Flight flight-id does not exist
```

Otherwise output the line

```
Passengers for flight flight-id
```

Then, output the passengers scheduled for this flight in alphabetic order, with last name first followed by a comma, one passenger per line. Number the passengers, with a "." following the number, starting the numbers at 1. The number should precede the passenger name. After all passengers have been output, output the total amount of money earned by the airline on this flight. If no passengers are scheduled for this flight your program should output

```
none
```

on the second line of this output and output

```
$0
```

on the third line.

4 city1 city2 — Find and output all pairs of flights where the first flight leaves `city1` and arrives in an intermediate city, and then the second flight leaves the intermediate city and arrives in `city2`. There must be at least 30 minutes between the arrival and departure in the intermediate city. The first line of output should be

```
One stop flights from city1 to city2
```

Then search for pairs of flights. The output, ordered by departure time from `city1`, must include three lines for each flight pair:

```
flight-id city1 departure-time intermediate-city arrival-time
flight-id intermediate-city departure-time city2 arrival-time
total-cost
```

where total cost is the sum of the costs of the two flights. The order of output, to make things simple, should be by the flight id of the first flight. When there are multiple flights using the same first flight, these should be ordered by the id of the second flight. Finally, if there are no such pairs of flights output the line

```
none
```

5 — quit the program, with no further output.

Please output a blank line between the responses to each input request. Example input and output will be provided on the course web site.

Important Notes

- Follow the input and output format specified above **precisely**. This will make the TAs' job of grading easier, ensure that students receive much more feedback on their code. If you are unsure of exactly what to output, follow the example from the course web site.

- You may use any technique discussed in class or covered in the text, up through and including maps and sets. You must use at least one map, however! You may also use any example code. In particular, you may find helpful the `Name` class from Lecture 11 (now split into `Name.h` and `Name.cpp`) and we have provided a simplified `Time` class, including an output operator, on the course web site.

- In all cases, check for errors in the order specified above. Whenever an error is found, output the error message, and then proceed to processing the next request (next line of input) rather than checking for more errors. In other words, you don't have to find all problems with each input line, just the first.