Due Date: 13.12.2020 at 23.55
Form of Delivery: You should upload a single zip filewith your solution on ODTUClass. The filename should be *EE441_PA2_firstname_lastname_studentID.zip*. Also indicate how much time you spent for the homework (see Regulations).

Important Notice— The following programming assignment requires theoretical knowledge on binary search trees, bidirectional graphs and further methods on these data structures. Should you have a lack of experience on the subject, it is highly recommend that lecture notes are revisited beforehand.

EE Airlines

EE Airlines is a newly established local air travel company that operates between the airports of Republic of Eeland. The employees require certain software to conduct tasks of recording flight routes, searching for them, establishing new ones and discarding those that are unprofitable. Besides, additional software is required for a 3rd parties, namely airport staff and travel agencies for scheduling, pricing, canceling of flights etc. In the homework, we will use the following important definitions:

- **Route:** directed one-hop connection between cities as represented in the route map
- **Flight:** directed one-hop connection between cities with associated flight number in the flight plan
- **One-way trip:** directed one- or multi-hop connection between cities (can require multiple flights)

You are now required to address the following tasks:

a) The flight route map of EE airlines consists of the following routes illustrated in the map given below. For their very first month on the market, the initial plan, along with the route fares for the corresponding routes, needs to be archived.

To Do: *Define a class that implements the directed graph data structure including an array of strings that stores the city names with available routes. The class should at least have the following methods*

- ADDGCITY: Add a city name at its corresponding location in the array of city names
- REMOVEGCITY: Remove a city name from the array of city names
- SEARCHGCITY: Search a city in the array of city names. Return its number if found and a default value otherwise
- ADDGROUTE: Add a route from a given departure city to arrival city with a specified route fare
- REMOVEGROUTE: Remove a route from a given departure city to arrival city
- SEARCHGROUTE: Search for a route from a given departure city to arrival city. Return the route fare if found.
- LISTGROUTES: Display the existing routes with fares in the command window

Note that the city names in Eeland are provided in the **citylist.txt** *file, which you can read into an array using the code given in the Appendix. Using your class, implement the global function* INITROUTEMAP *that initializes an instance of your class with the routemap shown in the figure. Route fares for each route can be determined using a random function.*

b) The monthly flight plan, derived from the route map in part a), is to be listed in order to allow the airport staff to use in boarding activities. Flight numbers are determined as 6-character strings, EEWXYZ first two of which is fixed as the company initials. WX is the numerical equivalent of the code of the city of departure, YZ is that of the city of arrival.

```
Ex: EE1506 is a flight from Signale (City 15) to East Circuitta (City 6).
```

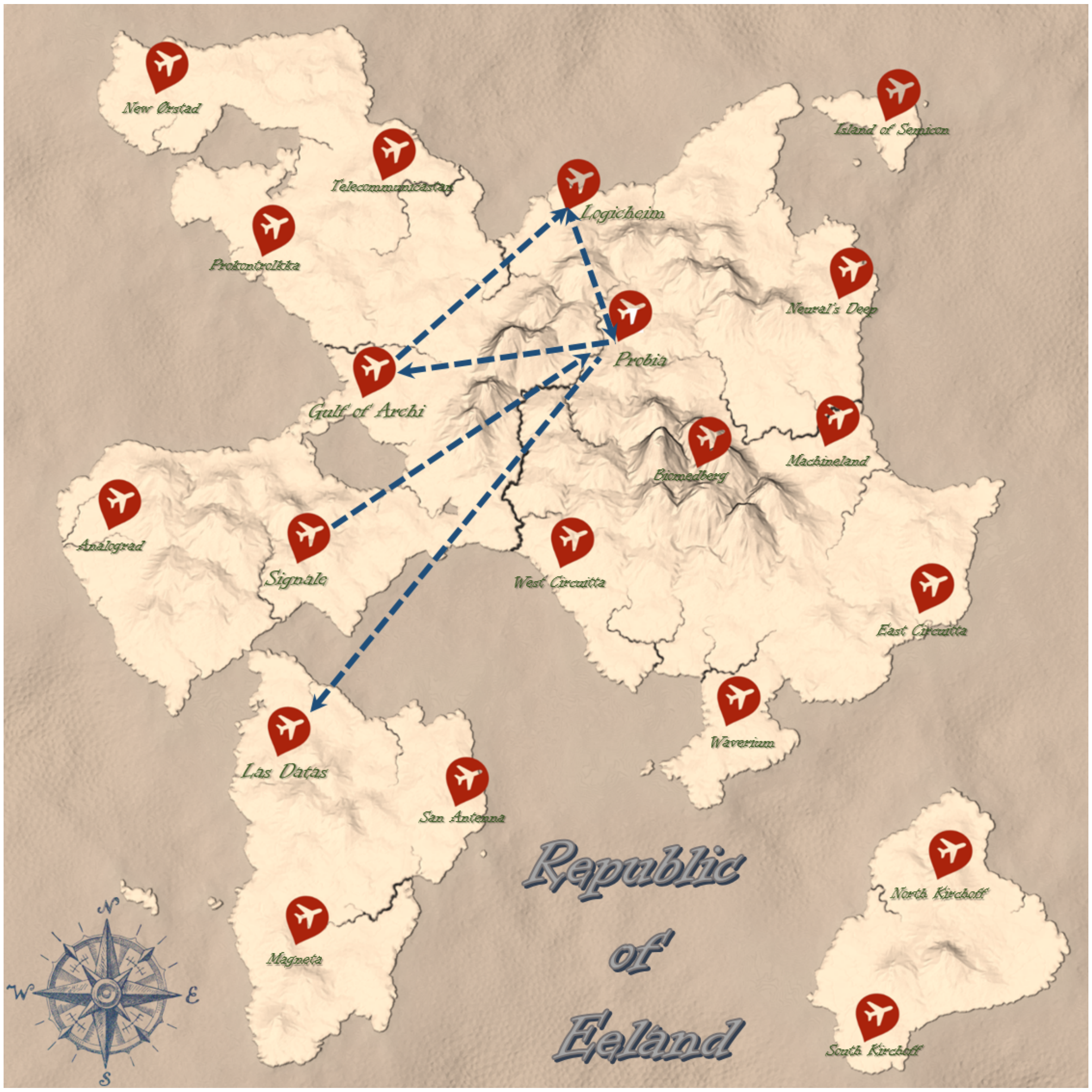

Fig. 1. Map of Republic of Eeland

To Do: *Define and implement a node class for a binary search tree that can store the flight information. Then, implement the global function* InitFlightPlan *that uses instances of your node class to build up a binary search tree with the flight plan based on the route map in a).*

Hint: *The node values can either contain int values (WXYZ) or string values ('EEWXYZ') as explained in the example. Note that implementing global methods such as* AddTFlight, RemoveTFlight, SearchTFlight *may help in this and upcoming tasks*. Think about the arguments and return values of these functions in analogy to part a).

c) The company offers N new routes starting form the beginning of each month; hence, these new routes need to be added – first to the existing route map, then to the existing flight plan.

To Do: *Implement the global function,* AddMonthlyNewRoutes *in which you update the routemap and print the newly added route information. In addition,* AddMonthlyNewRoutes *returns an array with the added flight numbers. Then, implement* AddMonthlyNewFlights*, another global function that adds the monthly new flights to the* *binary search tree in b) and prints the added flight numbers.*

Hint: *You may determine new routes by randomly selecting the place of departure and place of arrival from the list provided to you. Then, determine whether the random selections are already existing or not. N (int) can be given as an input to the function. City names in Eeland are provided in the* **citylist.txt** *file, which you can read into an array using the code given in the Appendix. You can test your function by trying different values of N.*

d) The travel agencies keep the passenger occupancy rates as feedback to airway companies. Using this information at the beginning of each month, EE Airlines halves the ticket prices for flights with low occupancy rate (lower than threshold OT%) in the preceding month and cancels flights with low occupancy rate in consecutive 2 months. Then, these routes are removed from the company's route map.

To Do: *Add a new private field,* OCCUPANCYRATE *and the public methods* SETOCCUPANCYRATE*,* GETOCCUPANCYRATE *to the node class in b) to set and get the occupancy rate of a given flight number. In addition, implement the following two global functions*

- CANCELUNPOPULARFLIGHTS: Remove flights from the binary search tree in b) as described above depending on the input argument OT (int). In addition, return (an) array(s) with the unpopular flight numbers and the canceled flight numbers.
- DISCARDUNPOPULARROUTESANDUPDATEPRICES: Use the outputs of CANCELUNPOPULARFLIGHTS to update the routemap in a).

These functions are also expected to print the information corresponding to price update and flight removal.

Hint: A random function will be more than enough to be used inside the "set" method to calculate monthly occupancy values. You can compare the old (before setting) and new (after setting) values of the occupancy rate to determine if a flight should be canceled. Ticket prices correspond to the route fares on the graph, and they will be updated accordingly.

e) The route map and the monthly flight plan needs to be checked by the planners, every 6 months, such that each route has a return connection and each flight has a return flight. If not, those lacking return need to be added and advertised.

To Do: *Implement two global functions:*

- CHECKNADDRETURNROUTES: Check the routemap in a) to determine which routes are missing. Then, add the routes and return an array with the new flight numbers.
- ADDRETURNFLIGHTS: Use the array returned by CHECKNADDRETURNROUTES to update the binary search tree in b).

The corresponding information (added routes and flights) should be printed.

f) The travel agencies satisfy the customer demands by offering them different one-way trip options, namely, direct, 1-stop, 2-stop.

To Do: *In order to address this requirement, implement a global function* DISPLAYROUTEOPTIONS. *The function receives the routemap object, the root of the binary search tree, the departure city and the arrival city (both as string (city name) or int (city code) value) as arguments. If at least one route between the entered cities is found in the routemap, the function displays the possible routes in the command window in the following form (as an example) and returns true:*

```
Direct Trip:
Probia -> EE0107 -> Logicheim, Fee: "computed fee"
1-stop Trips:
Probia -> EE0118 -> Gulf of Archi -> EE1807 -> Logicheim, Fee: "computed fee"
```

If no route exists between the entered cities, the function should display "There is no available route" and return false.

g) EE Airways publishes its monthly route map and flight report after all the updates. Provide a semi-annual (6-month) report similar to the "example run for PA2" on odtuclass (display on the command window is sufficient).

Hint: *The report will be automatically generated with the above implemented functions if you write a main function that applies the implemented classes and methods in the correct sequence for 6 months. N=5, as the monthly new route and OT=70% as the occupancy threshold can be used as example values. You may refer to the example report given in* ***example_run_PA2.txt*** *file.*

Regulations

- You should insert comments to your source code at appropriate places without including any unnecessary detail. **Comments WILL be graded.**
- Submit the whole Code::Blocks project folder in a zip file. The file name should be *EE441_PA2_firstname_lastname_studentID.zip*
- Indicate how much time you spent for the homework in a comment at the top of your main.cpp.
- Late submissions are welcome, but penalized according to the following policy:
  - 1 day late submission: HW will be evaluated out of 70 points
  - 2 days late submission: HW will be evaluated out of 50 points
  - 3 days late submission: HW will be evaluated out of 30 points
  - 4 or more days late submission: HW will not be evaluated

APPENDIX

The code to extract city names from the citylist.txt into a string array is provided below.

```
#include <fstream>
#include <string>
using namespace std;

int main () {
  string line;
  string city[20];
  ifstream myfile ("citylist.txt");
  if (myfile.is_open())
  {
   int i=0;
    while ( getline (myfile,line) )
    {
      city[i]=line;
      cout << city[i] << '\n';
      i++;
    }
    myfile.close();
  }

  else cout << "Unable to open file";

  return 0;
  }
```