CS 202, Summer 2021
Homework 4 — Hashing and Graphs

Due: 23:55, August 2, 2021

Important Notes

Please do not start the assignment before reading these notes.

a) Before 23:55, August 2, upload your solutions in a single ZIP archive using Moodle submission form. Name the file as `studentID_hw4.zip`.

b) Your ZIP archive should contain the following files:

- `hw4.pdf`, the file containing the answers to Questions 1 and 2.
- All C++ source files (`FlightMap.h`, `FlightMap.cpp`, and `main.cpp`) as well as any additional class source code and header files, and the `Makefile` for Question 3.
- `readme.txt` the file containing anything important on the compilation of your program in Question 3.
- Do not forget to put your name and student id in all of these files. Well comment your implementation. Add a header as given below to the beginning of each file:

```
/*
 * Title: Hashing and Graphs
 * Author: Name Surname
 * ID: 21000000
 * Section: 1
 * Assignment: 4
 * Description: description of your code
 */
```

- Do not put any unnecessary files such as the auxiliary files generated from your favorite IDE. Be careful to avoid using any OS dependent utilities (for example to measure the time).
- You should upload handwritten answers for Q1 and Q2 (in other words, do not submit answers prepared using a word processor).
- Use the exact algorithms shown in lectures.

c) Although you may use any platform or any operating system to implement your algorithms and obtain your experimental results, your code should work on the `dijkstra` server (dijkstra.ug.bcc.bilkent.edu.tr). We will compile and test your programs on that server. Thus, you will lose a significant amount of points if your C++ code does not compile or execute on the `dijkstra` server.

d) This homework will be graded by your TA, Berat Biçer. Thus, please contact him directly (berat.bicer at bilkent.edu.tr) for any homework related questions.

Attention: For this assignment, you are allowed to use the codes given in our textbook and/or our lecture slides. However, you ARE NOT ALLOWED to use any codes from other sources (including the codes given in other textbooks, found on the Internet, belonging to your classmates, etc.). Furthermore, you ARE NOT ALLOWED to use any data structure or algorithm related function from the C++ standard template library (STL).

Do not forget that plagiarism and cheating will be heavily punished. Please do the homework yourself.

Question 1 — 24 points

a) [6 points] Assume that we have a hash table with size 13 (index range is $0 \ldots 12$), and we use the `mod` operator as a hash function to map a given numeric key into this hash table. Draw the hash tables after the insertion of the keys

```
12 15 20 30 41 29 17 25 22
```

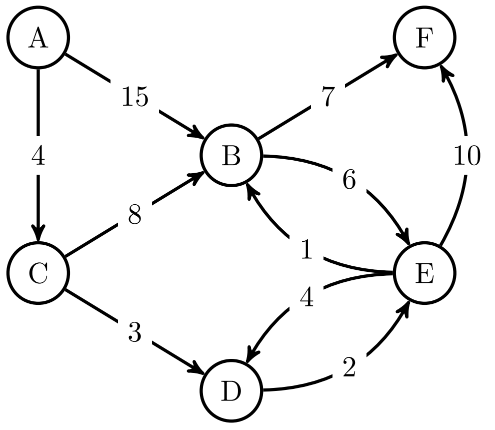
in the given order for each of the following methods:

- open addressing with linear probing,
- open addressing with quadratic probing,
- separate chaining.

b) [18 points] Find the average number of probes for a successful search and an unsuccessful search for each of the hash tables that you created in part a.

Question 2 — 6 points

For the following weighted directed graph, find the shortest paths from vertex A to all other vertices using Dijkstra's shortest path algorithm. Show all steps of Dijkstra's algorithm in your solution.



Question 3 — 70 points

You are asked to develop a program that allows users to do some operations on a flight map that is represented as an <u>**unweighted directed graph**</u> where the vertices correspond to the cities and the edges correspond to the flights between these cities. These operations should be implemented in a class named `FlightMap` with all related code in the files named `FlightMap.h` and `FlightMap.cpp` (as well as any other files that contain any extra class that you wrote). The class definition and the descriptions of the required public functions are given below.

```
class FlightMap {

 public:
   FlightMap(const string vertexFile, const string edgeFile);
   ~FlightMap();

   void findDirectFrom(const string source);
   void findDirectTo(const string destination);
```

```
    void findPath(const string source, const string destination);
    void findPath(const string source, const string destination, const int stops);
    void findShortestPath(const string source, const string destination);

    ...
  private:
    // define your data members here
    // define private member functions here, if any
    // you MUST use the adjacency list representation
};
```

`FlightMap(const string vertexFile, const string edgeFile);` The constructor loads the flight map from the given text files containing the vertices and the edges. The first input is a text file named `vertexFile` that contains each city in a separate line. The second input is a separate file named `edgeFile` that contains the edge information that represents the flights. Each line includes an edge with the cities separated by a comma. The first city is the source city and the second one after the comma is the destination city.

The constructor must load the map data from these files and construct an **adjacency list** representation for the graph. You MUST use the adjacency list representation in this assignment. (Otherwise, you will get no points from this question). Note again that the edges are directed edges with no weights.

You MUST also design a **hash table** to implement the mapping from the cities to unique integer indices. These integer indices (in the range from 0 to N-1) where N is the number of cities are used to refer to the vertices (i) and edges (i,j) internally in the code whereas the input and output use the actual city names. The hash table takes a city name (string) as input and gives its index (integer) as the output. The goal of the hash table is to perform this lookup efficiently. You MUST use the hash function (Hash Function 3) given on slide 40 of the lecture notes on hashing. You are free to choose the table size as you wish. The hash table MUST use **open addressing with quadratic probing** to store the data. The integer indices will correspond to the line numbers for the cities given in the vertex file. You must use this hash table with the city names given as the keys and the line numbers as the item data that will be used as the vertex indices.

`void findDirectFrom(const string source);` This function lists all destination cities that have a direct flight from this source city. Each destination city should be displayed on a separate line in the output. Display a warning message if the input city does not exist in the graph.

`void findDirectTo(const string destination);` This function lists all source cities that have a direct flight to this destination city. Each source city should be displayed on a separate line in the output. Display a warning message if the input city does not exist in the graph.

`void findPath(const string source, const string destination);` This function finds and prints all paths from the source city to the destination city. Each city on each path should be displayed on a separate line in the output. Each path should be clearly separated from the others. Display a warning if any one of the input cities does not exist in the graph.

`void findPath(const string source, const string destination, const int stops);` This function finds and prints all paths from the source city to the destination city with the given number of stops. The source and destination cities are not counted as stops in this computation. Each city on each path should be displayed on a separate line in the output. Each path should be clearly separated from the others. Display a warning if any one of the input cities does not exist in the graph.

`void findShortestPath(const string source, const string destination);` This function displays the cities along the shortest path from the source city to the destination city. If there are multiple such paths, displaying one of them is sufficient. Each city on the path should be displayed on a separate line in the output. Display a warning if any one of the input cities does not exist in the graph.

No flight map data are provided as part of this assignment. That is, you are responsible from generating example text files to test your own solution. You must submit your `main` function in a file named `main.cpp` together with example flight map data files that you generated as test files as part of your solution.