Centre for infrastructure, Sustainable Transportation and Urban Planning

Indian Institute of Science (IISc), Bengaluru
C++ Developer: Round 1

The aim of this exam is to test your problem-solving skills and basic understanding of C++. All questions are NOT compulsory, and partial solutions will also be considered while shortlisting for subsequent rounds. Hence, you are encouraged to submit the best possible answer for each question. Follow the instructions below precisely.

- Plagiarism will result in instant disqualification. You must write your own code.
- The test requires you to CLONE a GitHub repository and work on it. To submit your solution, follow the steps:
  - Push the updated/new files to your repository and add Prateek1009 (link) as collaborators. Make sure your repository is private. Public repositories will not be considered for evaluation.
  - Fill the following Google Form: link. You are allowed to make only one submission.
- The test commences on 22^{nd} March 2024 (10:00 AM). The last date for submission is 25^{th} March 2024 (05:00 PM). Late submissions will not be accepted.
- If selected, it will be mandatory for you to join in an in-person capacity. Please refrain from attempting the test if you cannot attend in-person.
- The test will be used to shortlist for "Network Optimization Models for Traffic and Transit (C++ Developer)", supervised by Prateek Agarwal and Dr. Tarun Rambha.
- For queries, contact prateeka@iisc.ac.in

All the best.
C*i*STUP
Indian Institute of Science, Bengaluru

Question 1 (50 Marks)

You will be working with the GitHub repository "ULTRA: UnLimited TRAnsfers for Multimodal Route Planning"(link), which contains various state-of-the-art shortest-path algorithms. Additionally, you have been provided with a graph of Florida in the DIMACS format (link). Below is a summary of the DIMACS format. For more details, refer to link.

```
c file: TestCase1-Sample test case for C++ Developer position
c source: Prateek Agarwal (prateeka@iisc.ac.in)
p sp 4 6
a 1 2 5
a 1 4 1
```

- Comments. Lines starting with "c" are comments, providing human-readable information and are ignored by programs.
- Problem line. The line starting with "p" specifies the problem type ("sp" for shortest path), the number of nodes "n", and the number of edges "m" in the graph.
- Edge Descriptors. Each edge is described using "a u v w", indicating an edge from node "u" to node "v" with weight "w".

Your task is to create a file named `question1.cpp` inside the `Runnables` folder. When compiled, `question1.cpp` runs Dijkstra's algorithm implemented in the repository for 200 randomly selected source-destination pairs on the Florida graph. Note that `question1.cpp` calls Dijkstra's algorithm already implemented in the repository. Do not code your own Dijkstra's algorithm. `question1.cpp` prints the total runtime in seconds in the following format:

`Total runtime in seconds for 200 random Dijkstra:  x`

Hints:

- Here are a few test cases for you to verify the Dijkstra's output:
  - Source: 264345, Target: 264327
    Shortest Path: 264345 - 269065 - 264331 - 264311 - 264324 - 264327
    Shortest path length: 10283
  - Source: 1234, Target: 1272
    Shortest Path: 1234 - 1235 - 1228 - 1226 - 1238 - 1247 - 1265 - 1272
    Shortest path length: 11158
  - Source: 0, Target: 1
    Shortest Path: 0 - 1
    Shortest path length: 14188
- A node labeled x in the DIMACS graph, will be labeled *x-1* in the custom binary format. For example, a node labeled 1 in the DIMACS graph will be represented as node 0 in the custom binary format. The sample cases provided above are w.r.t. custom binary format.
- You can spend weeks understanding all the algorithms. However, the question expects you to identify how and where Dijkstra's algorithm is used in the codebase and replicate the process. The question can be completed in less than 25 lines!

Question 2 (50 Marks)

Modern routing platforms like Google Maps use routing engines to plan passenger journeys. These engines use timetable information in the GTFS format (refer reference guide) to find possible journeys between a starting point and a destination. In this context, a "stop" refers to a specific location where public transportation vehicles, such as buses or trains, pick up or drop off passengers. Each stop is identified by a unique stop ID, and a "route" represents a specific path taken by a transportation vehicle.

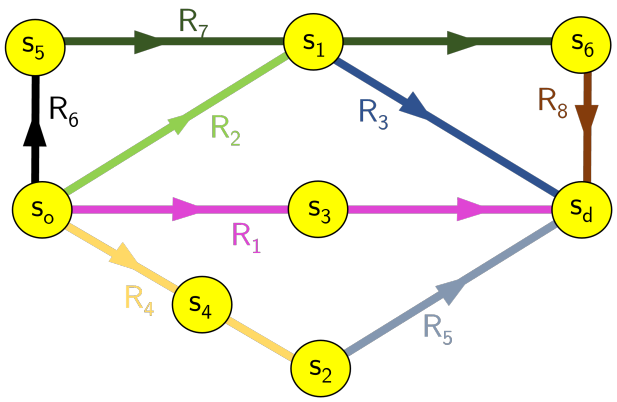

Figure 1: Toy Network

Consider the toy bus network with stops shown as yellow nodes and routes represented by solid lines. For example, the route R_1 in pink connects three stops s_o, s_3, and s_d. Your task is to write a C++ program that finds the following:

- All "direct journeys" connecting the source stop (s_o) and destination stop (s_d). E.g., Route R_1.
- All journeys with one transfer connecting the source stop (s_o) and destination stop (s_d). E.g., follow route R_2 from s_o to s_1 and then take route R_3 from s_1 to s_d.

Problem Statement: Write C++ code that reads the input GTFS data (input data) and takes two command-line inputs, the source stop ID and the destination stop ID. The code should print all possible direct journeys and journeys with one transfer. In the above example, the output should look like this:
Direct journeys: $R_1(s_o > s_d)$
Journeys with one transfer: $R_2(s_o > s_1) - R_3(s_1 > s_d)$, $R_4(s_o > s_2) - R_5(s_2 > s_d)$.

Submission steps: To submit the solution, create a folder named "`Question 2`" in your GitHub repository and place all related codes/ input data inside it.

Bonus (25 Marks): Print journeys with two transfers also. For example, $R_6(s_o > s_5) - R_7(s_5 > s_6) - R_8(s_6 > s_d)$

Hint: This question assesses your algorithmic skills and problem-solving ability. If selected, you will be working on similar problems. For reference on GTFS data, refer link. You can make suitable assumptions to complete the question.