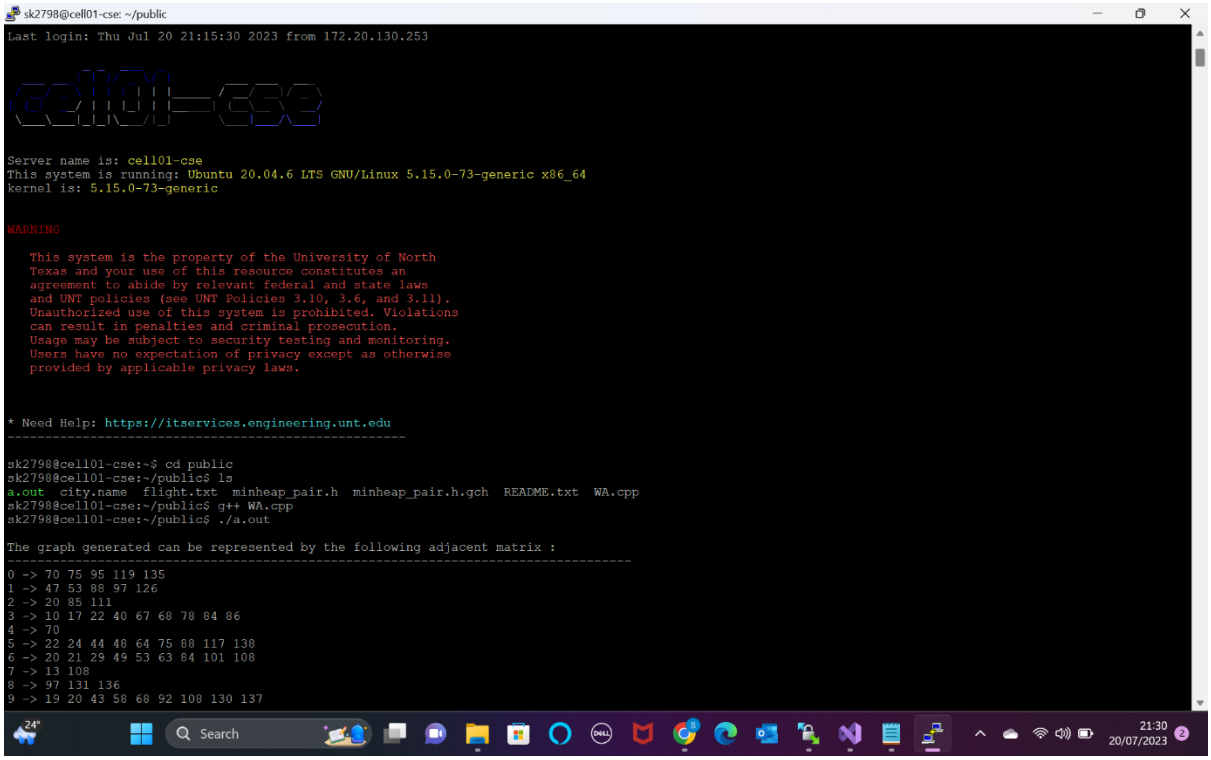
Assignment 4

Code Implementation:

1. Compilation Process:

The instructions that are used to compile the code are:



- >>cd public-command is used to change the current directory to the public directory. By doing this, you can view and interact with files in the "public" directory without constantly giving the complete path.
- >>ls-It will list all the files present in the public folder.
- >>g++ WA.cpp- is used to compile the code
- >>./a.out- is used to run the code

2. Code Description:

The routeSearch_2 function implements Task 2, which seeks to determine the route from city "A" to city "D" through cities "B" and "C" with the fewest connections. The graph is the class that represents the graph, and city_A, city_B, city_C, and city_D are integer IDs of the cities in the graph. The function is declared with the signature void routeSearch_2(Graph graph, int city_A, city_B, city_C, and city_D).The function begins by setting up a few variables:

The number of cities in the graph overall, as determined by graph.get().

The Dijkstra algorithm uses the following arrays to hold the shortest distances and parent nodes from city "A" to cities "B" and "C," accordingly.

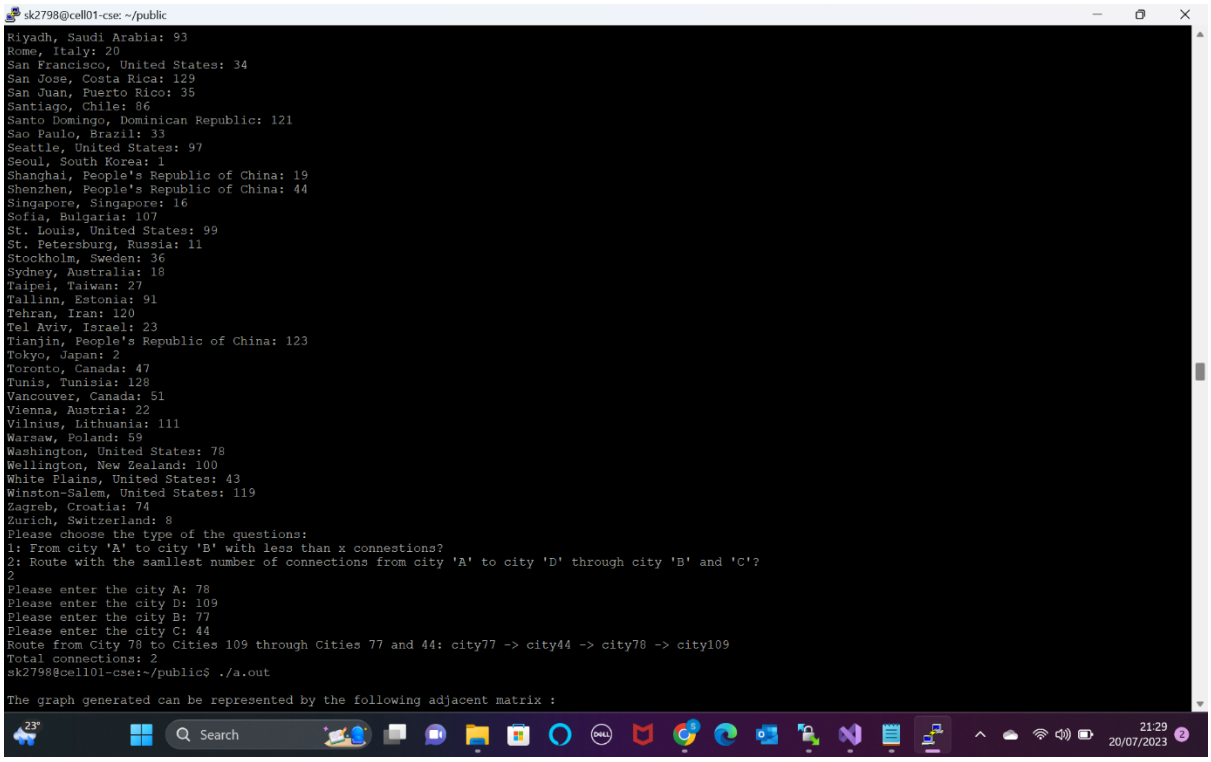
d_CD[n], p_CD[n]: Arrays to hold, throughout the Dijkstra algorithm, the shortest distances and parent nodes from cities "B" and "C" to city "D," respectively.

The ultimate pathways from city "A" to cities "B" and "C," and from cities "B" and "C" to city "D," are each stored in a vector, ans_AB and ans_CD.In order to determine the shortest routes from city "A" to cities "B" and "C" (using Dijkstra(graph, city_A, d_AB, p_AB)) and from cities "B" and "C" to city "D" (using Dijkstra(graph, city_D, d_CD, p_CD)), the function makes two calls to the Dijkstra function. There may be routes from city "A" to city "B" and from city "B" and "C" to city "D," respectively, if the two shortest distances, d_AB[city_B] and d_CD[city_C], are not equal to INT_MAX.

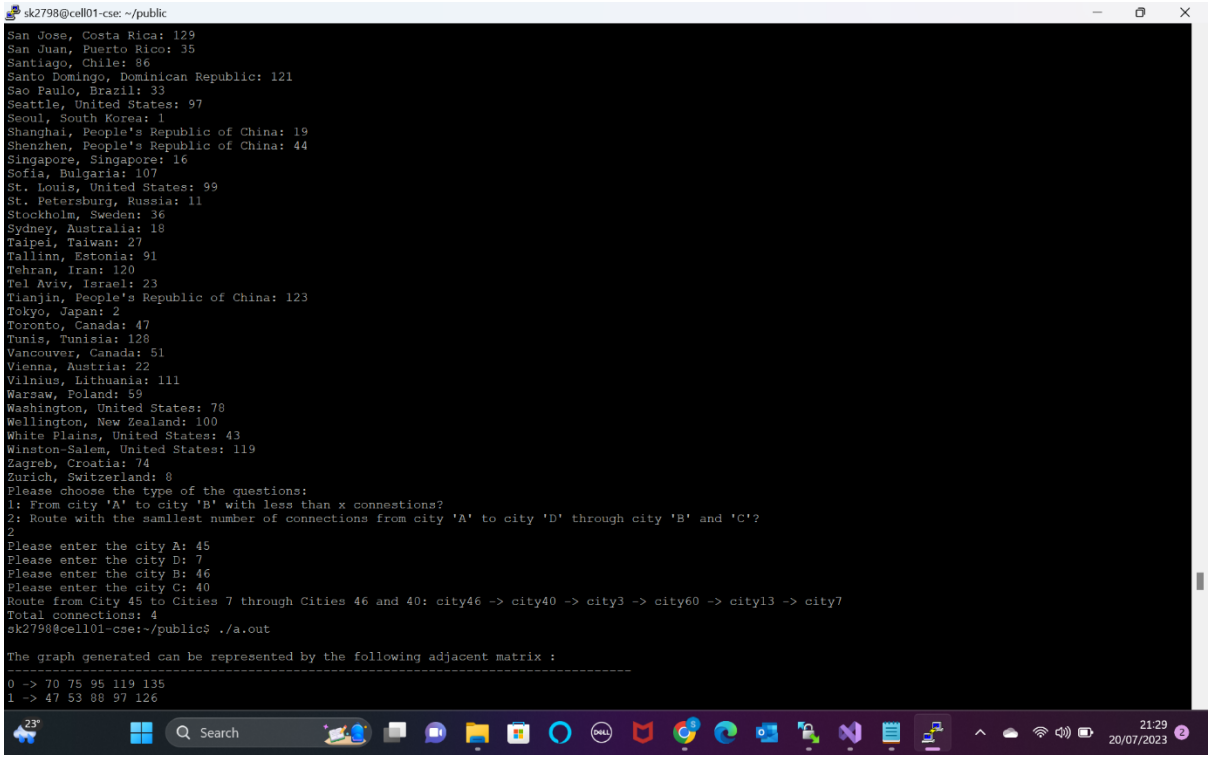
After that, the method reconstructs the routes from city "A" to city "B" and from cities "B" and "C" to city "D" using the parent arrays p_AB and p_CD.

The two paths (ans_AB and ans_CD) are combined into the final route by inserting the items from ans_CD after ans_AB once the paths have been rebuilt. The final output of the function includes the total number of connections along with the ultimate route from city "A" to city "D" via cities "B" and "C." If any of the shortest distances, d_AB[city_B] or d_CD[city_C], is INT_MAX, then there isn't a path connecting city "A" and city "D" via cities "B" and "C." In this situation, the function outputs "No such route."

3. Output:



In this case, the shortest route from city 78 to city 109 goes through cities 77 and 44, with a total of 2 connections between the cities. The output provides a clear representation of the route and the number of connections taken to reach the destination city 109 from the starting city 78 through cities 77 and 44. This line shows the path that travels from city 78 to city 109, and it specifies that cities 77 and 44 are on the path. The order of the cities along the route is depicted by this line. The path departs from city 77 and travels to city 44, city 78, and lastly city 109.



In these two screenshots, the shortest distance obtained is 4 connections and for the other connections route is not found.