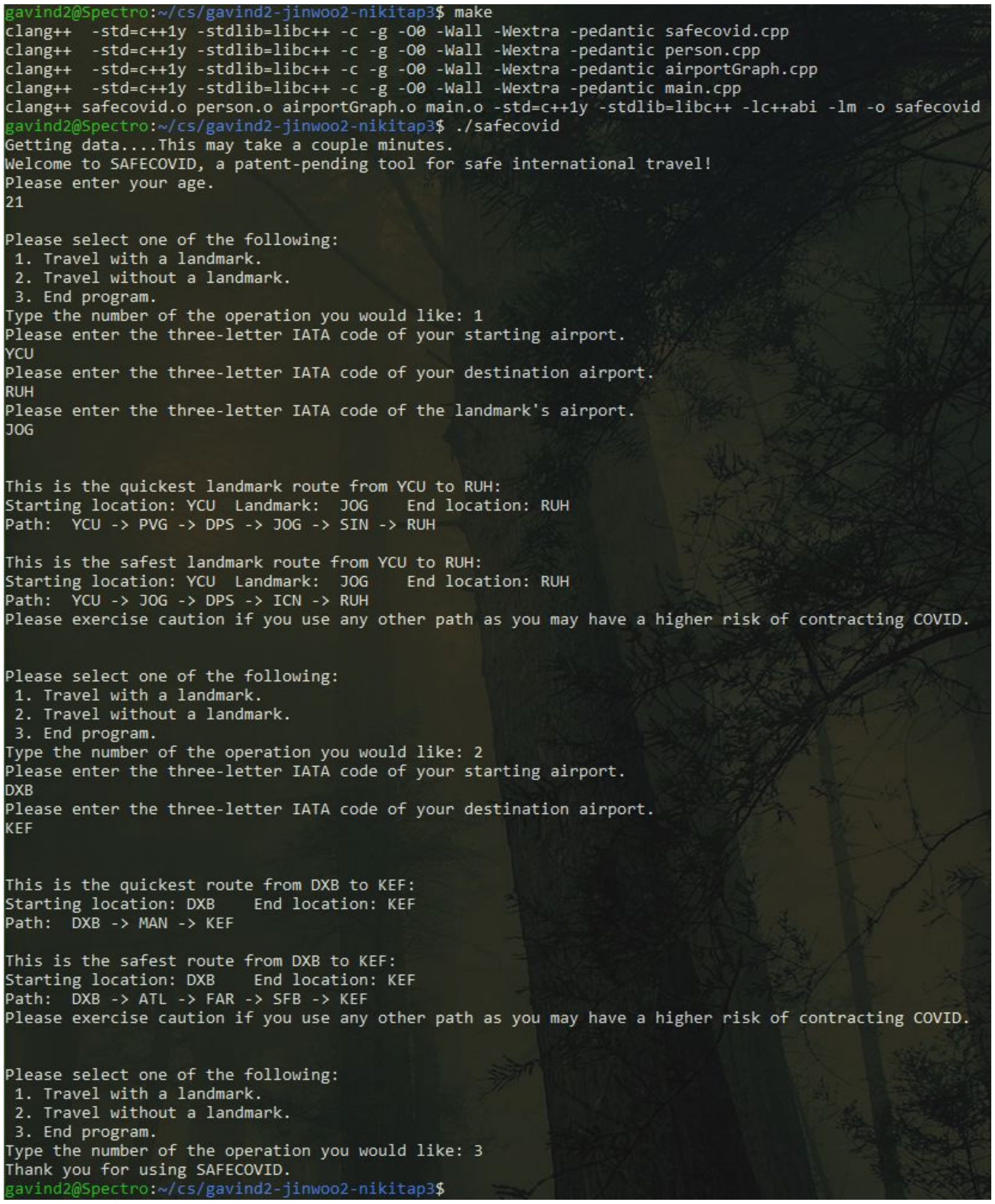
gavind2, jinwoo2, nikitap3

SAFECOVID Results:

When SAFECOVID runs, it imports the text file containing the airports as nodes and the airlines as edges, found in our “data” folder as “edges.txt”. Then, the program starts a user interface where it prompts the user to select one of three options: 1) view the path from a departure location to a destination of their choice with a landmark to stop by, 2) view a path from a departure to destination without the landmark, or 3) end the program. When either of the first two selections is chosen, the program uses a BFS and Dijkstra’s algorithm to find the quickest and optimal path. The optimal path is determined by the rate of which the person may be exposed to the virus via the number of edges in each node. Each edge is weighted and when each edge is used, the rate is summed based on the edge’s weight. When using a landmark, the same procedure takes into action, except it is run twice: from start to landmark, and landmark to destination. Once the run is complete, the program returns to where the user selects which action to take due to the switch-case loop. The program concludes when the user selects the last option that exits the program. Figure 1 at the end shows the terminal when we run the program for two different scenarios.

In the end of developing this project we met most of our goals. We reused the graph implementation from lab_ml, and filled this graph with vertices and edges from the dataset which contains the routes between different airports. We implemented BFS, Dijkstra, and Landmark algorithms to get paths between two locations with and without support of landmarks. Using these algorithms we were able to get the shortest path (done with BFS) and the safest path (done with Dijkstra’s algorithm). However, we did not plan to use Dijkstra initially. We wanted to implement cycle identification to ensure we don’t loop between the same few airports. However, in the end we decided that Dijkstra would be more useful. This is also due to the fact that Dijkstra’s algorithm does not work with negative-path cycles. Since we were able to implement this algorithm, we effectively did cycle prevention as well. We also added a user friendly console interface. Using this interface, we are able to run our program multiple times with different inputs without needing to remake our starting graph every single time. This saved minutes of run time. We also achieved another goal; we implemented a “person” class and a function to calculate an approximate chance of contracting COVID. This was based on the degree of each airport, as a higher degree implied more people and therefore a greater chance to contract COVID. This allowed us to find the safest route using Dijkstra algorithms. We tested most of our algorithms.

This project served to expose us to GitHub and version control. During this project, we created separate branches aside from our master branch. These separate branches, denoted by each person’s NetID, were “working branches”. Here, we could work on our code and have other team members review it before being merged into the clean and final “master” branch. However, the downside is that we needed to make sure all local and remote branches were always up to date. Not taking care of this led to many merging issues such as linker errors. More than once, we had to delete the files from our local branch and re-clone our repository. While this wasn’t fun work, we ended up understanding why it was beneficial to work like this, even though it took a lot of practice. We also got more exposure to deliberately coding in graphs, which is something that we unknowingly did for a few MP’s and had only done for one lab assignment. This provided us a better chance to experiment with graph implementations and show real-world applications of graphs and reading data.



```
gavind2@Spectro:~/cs/gavind2-jinwoo2-nikitap3$ make
clang++  -std=c++1y -stdlib=libc++ -c -g -O0 -Wall -Wextra -pedantic safecovid.cpp
clang++  -std=c++1y -stdlib=libc++ -c -g -O0 -Wall -Wextra -pedantic person.cpp
clang++  -std=c++1y -stdlib=libc++ -c -g -O0 -Wall -Wextra -pedantic airportGraph.cpp
clang++  -std=c++1y -stdlib=libc++ -c -g -O0 -Wall -Wextra -pedantic main.cpp
clang++ safecovid.o person.o airportGraph.o main.o -std=c++1y -stdlib=libc++ -lc++abi -lm -o safecovid
gavind2@Spectro:~/cs/gavind2-jinwoo2-nikitap3$ ./safecovid
Getting data....This may take a couple minutes.
Welcome to SAFECOVID, a patent-pending tool for safe international travel!
Please enter your age.
21

Please select one of the following:
 1. Travel with a landmark.
 2. Travel without a landmark.
 3. End program.
Type the number of the operation you would like: 1
Please enter the three-letter IATA code of your starting airport.
YCU
Please enter the three-letter IATA code of your destination airport.
RUH
Please enter the three-letter IATA code of the landmark's airport.
JOG


This is the quickest landmark route from YCU to RUH:
Starting location: YCU  Landmark:  JOG    End location: RUH
Path:  YCU -> PVG -> DPS -> JOG -> SIN -> RUH

This is the safest landmark route from YCU to RUH:
Starting location: YCU  Landmark:  JOG    End location: RUH
Path:  YCU -> JOG -> DPS -> ICN -> RUH
Please exercise caution if you use any other path as you may have a higher risk of contracting COVID.


Please select one of the following:
 1. Travel with a landmark.
 2. Travel without a landmark.
 3. End program.
Type the number of the operation you would like: 2
Please enter the three-letter IATA code of your starting airport.
DXB
Please enter the three-letter IATA code of your destination airport.
KEF


This is the quickest route from DXB to KEF:
Starting location: DXB    End location: KEF
Path:  DXB -> MAN -> KEF

This is the safest route from DXB to KEF:
Starting location: DXB    End location: KEF
Path:  DXB -> ATL -> FAR -> SFB -> KEF
Please exercise caution if you use any other path as you may have a higher risk of contracting COVID.


Please select one of the following:
 1. Travel with a landmark.
 2. Travel without a landmark.
 3. End program.
Type the number of the operation you would like: 3
Thank you for using SAFECOVID.
gavind2@Spectro:~/cs/gavind2-jinwoo2-nikitap3$
```

Figure 1: Terminal output after running our program