Name: Anurag Reddy Pingili

UTA ID: 1001863623

**Code used**: Java

**Requirements:**

Java Compiler which can be downloaded here: [Java Downloads for All Operating Systems](https://www.java.com/en/download/manual.jsp)

**Instructions:**

* Open a terminal and use cd command to set the directory to the Part1 file in the extracted folder.
* Use *javac find\_route.java* to compile the program.
* To run the program, use the command:

*java find\_route.java <text file> <Source City> <Destination City>*

for Uninformed search

and for Informed search use:

*java find\_route.java <text file> <Source City> <Destination City> <text file>*

* If insufficient or excessive number of arguments are entered, the program will instruct you to enter correct number of arguments.

**NOTE:** Names of cities and file names are case-sensitive. For example, ‘Hassel’ cannot be entered as ‘hassel’.

**Program Structure:**

The code takes the parameters: Text file, Source City and Destination City, to find the optimal route between the two cities.

1. The code checks if the correct number of arguments are given.
2. It reads the data from Input file and heuristic file and stored it in *city\_details* ArrayList.
3. *fringe* is initialized with the Source City and its *ParentNode* is set as *NULL.*
4. *findNeighbors()* method is called to find the child nodes of the current node in the fringe which is popped.
5. The popped node is added to *data.visit.*
6. The fringe is then sorted based on whether it is an Informed Search or Uninformed Search such that Uninformed Search uses only total cost but Informed Search uses total cost and heuristic value.
7. If there is no node in the fringe and there is no path from source to destination, infinity is returned.
8. It checks whether the current node popped is the destination and if it is, control exits *findNeighbors()* and *calculate\_path()* is called.
9. *calculate\_path()* finds the optimate path using the *data.visit* list and finds the path with the least cost.
10. *Nodes\_expanded* is used to store the number of nodes expanded and

*Nodes\_generated* stores the number of generated nodes.

**References:**

* [**https://github.com/topics/java-project**](https://github.com/topics/java-project)
* [**https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-in-java-using-priorityqueue/**](https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-in-java-using-priorityqueue/)
* [**https://stackoverflow.com/questions/tagged/java**](https://stackoverflow.com/questions/tagged/java)