TO MAP A CITY USING 3-D LINKED LIST

MTE-PROJECT REPORT

SUBMITTED AGAINST THE MTE COMPONENT OF THE COURSE

DATA STRUCTURES (CS-251)

SUBMITTED BY-

GUDDU KUMAR, 2K20/MC/52

HEMAANK MAHAJAN, 2K20/MC/55



COURSE TEACHER

**(Dr. Goonjan Jain)**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Shahbad Daulatpur Delhi-110042

NOVEMBER-2021

# DECLARATION

We, Guddu Kumar (2K20/MC/52) and Hemaank Mahajan (2K20/MC/55) students of B. Tech (Mathematics and Computing) hereby declare that the project report titled “To map a city using 3-D Linked List” which we submit to the Department of Computer Science and Engineering, Delhi Technological University in fulfillment of the requirement for the credits of CS-251, is original and not copied from any source without proper citation.

Place: Delhi Hemaank Mahajan

Date: 17-11-2021 Guddu Kumar

# CERTIFICATE

I hereby certify that the project report titled “To map a city using 3-D Linked List” submitted by Guddu Kumar (2K20/MC/52) and Hemaank Mahajan (2K20/MC/55) pursuing their Bachelors of Technology in Mathematics and Computing at Delhi Technological University is a record of work carried out under my supervision. As far as I know, this project is not under consideration for credits of any other diploma or degree at Delhi Technological University or elsewhere.

Dr. Goonjan Jain

SUPERVISOR

# ACKNOWLEDGEMENT

In performing our major project, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this assignment gives us much pleasure. We would like to express our gratitude to our professor Dr. Goonjan Jain ma’am who gave us the golden opportunity to do this wonderful project on the topic to map a city using 3-D Linked List, which also helped us in doing a lot of research and we came to know about a lot of new things, we are really thankful to her. In addition, we would like to thank Department Computer Science and Engineering, Delhi Technological University for giving us the opportunity to work on this topic.

# ABSTRACT

In our MTE project we have invented our own data structure i.e., 3-D Linked list. We have defined what is 3D Linked List and its different properties. We have made an animation for the same in Python turtle graphics. We have used Singly Linked List, Stack implemented as Linked List and array (for taking input only) in constructing of the grid. A base grid is always present in every application. We have defined many functions to work on the grid like insert, high data, display, delete high data, etc. keeping in mind that they have the smallest time complexity. The time complexity of all the functions is also discussed. Project is distributed in 3 versions according to their suitability in different applications. In version 1 we have plotted the map of a city by placing a grid on it and assigning key to every type of location. Additional information about that place is stored at its height by the use of high data function. More is the size of grid more accurately we can plot the city. For simplicity we have taken a grid of 15x15 i.e., 225 elements and high data is stored at selected locations. In version 2 we have constructed the shape of the lattice as a cuboid which further increases the flexibility to access a given data point. We only need the coordinates in 3 dimensions and we can access the data and last but not the least in version 3 we have applied the project on finding routes within a city by using 3D Linked List. Here, a finite number of paths are compared and the path with lower time is displayed along with its time.

# Contents

Title Page

Candidate’s declaration

Certificate

Acknowledgment

Abstract

Theory of Project

Version 1 (Mapping a city)

Key

Images of Map and Animation

Synopsis of function used

Version 2 (Lattice Structure-Cuboid)

Synopsis of function used

Version 3 (Locating Routes)

Synopsis of function used

Output Images

Conclusion

Reference

# THEORY OF THE PROJECT

3-D Linked List is a linked data structure where every node is connected up to 3 more nodes in such a fashion that they form a lattice.

In 3-D Linked List there is always a Base Grid which is made up of interconnected Nodes. Here, every node has the potential of connecting up to 2 more nodes in the base grid which can be one along the length and one along the breadth.

The base grid will thus lead to numerous connection due to which there can be many paths to reach a node, which provides flexibility to access data.

Every node in the base grid has the capacity to store data at its height and the data in the height is stored in the manner of stack implemented as linked list.

# MAPPING A CITY [VERSION 1]

In this version of the project, we have taken the map of a city and placed a grid on it.

A key is made in which different numbers will represent different locations and accordingly values are assigned to the nodes in the grid which is constructed dynamically.

Additional information about the places is stored at higher level or at height of the corresponding nodes.

## KEY

1-5 Describes land type

1 – Grass

2 – Cemented/Constructed areas

3 – Water/Ocean/lake

4 – Road

5 – Barren Area

21 - Residential Area

22 - Park

23 - Military area

24 - Government Office

25 - Village area

26 - Museum

27 - Airport

28 - Highway

29 - School

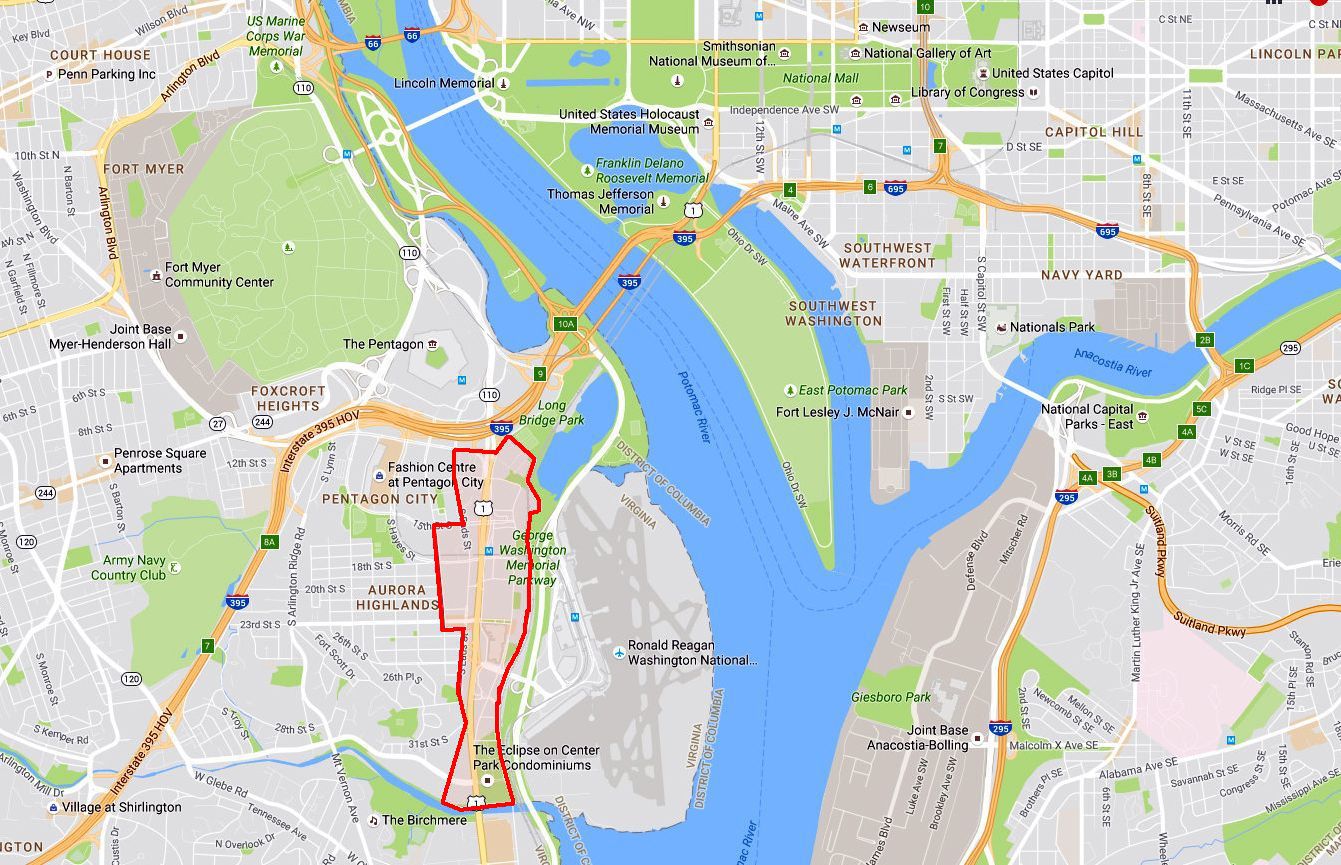
30 - Court

31 - Mall

32 – Office

Empty Spaces is given between keys [5-20] for future work.

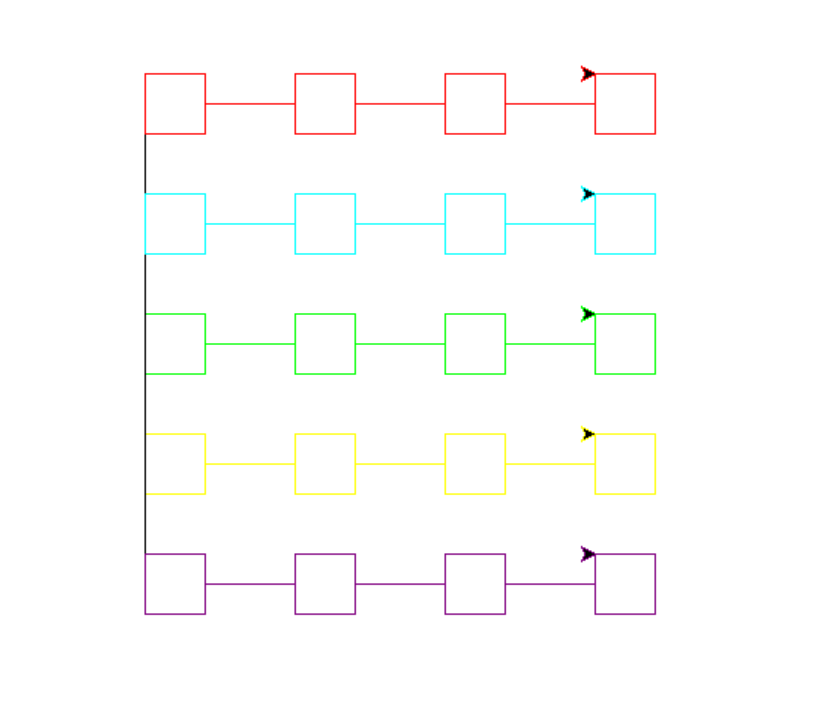
Here, in the project Crystal City is taken whose map is shown below.



After placing the grid, the map is shown below.



A graphic representation is also made in the project as a part of the presentation which is as shown below.



# SYNOPIS OF FUNCTIONS USED IN VERSION 1

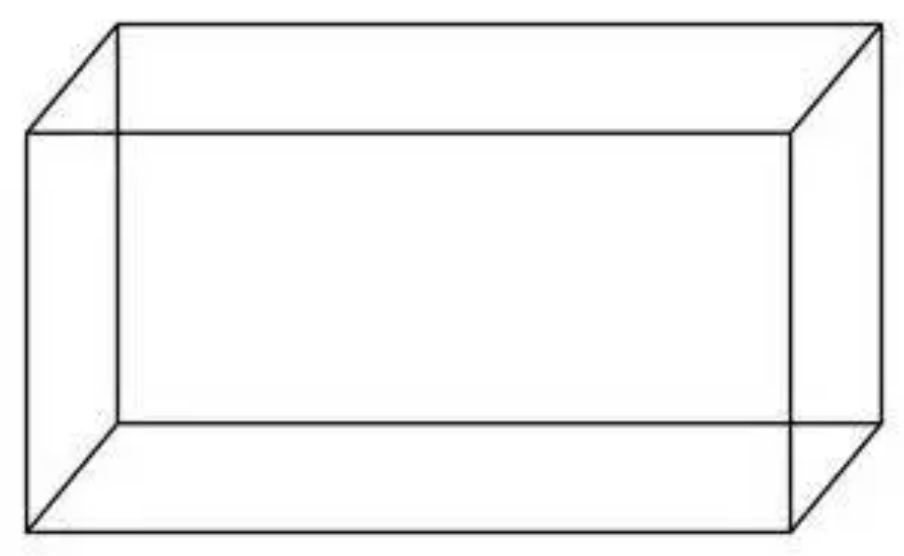
Insert Function – It constructs the grid and store the data in the given co-ordinates. Time complexity is O(n)

High data function – It is used when the data in the given coordinate is full and more data is required to be stored there. Time complexity is O(n).

Display function – It prints all the value of the given coordinate form data of its base grid to all data present above it. Time complexity is O(n).

Delete high data function – It is used to delete the high data present at the top most position of the node’s location. If no high data is present then it will free the node at the base grid. Time complexity is O(n).

# CUBOID LATTICE [VERSION 2]



Version 2 is different from version 1. In version 1 there was only a base grid where nodes where interconnected but the high data were only accessed through the node which was present at that location of the base grid.

In version 2 the structure of the lattice is cuboid, which means that there is a grid present at every level. So, accessibility of data will further increase.

All the grids thus formed will be connected by the height pointer of the corresponding nodes of the grid below.

Here a node can only store a limited amount of information which is as described in its data structure and not variable amount as it was in version 1. Hence both these versions have their own usage of their kind.

# SYNOPIS OF FUNCTIONS USED IN VERSION 2

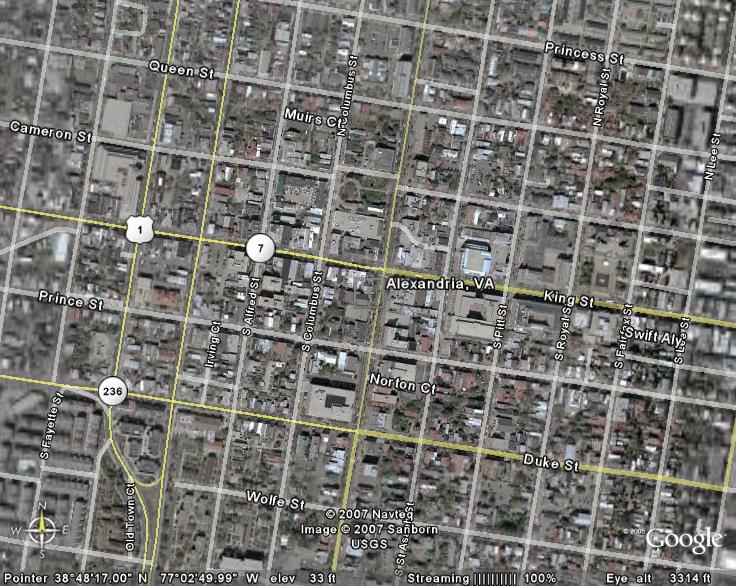
Insert function – It constructs the lattice and place the element at the given location in terms of the 3\_dimensional coordinates. Time complexity is O(n).

High data function – it is used to make the connection of the new node to the corresponding node in the below grid. Time complexity is O(n).

Display function – It prints the data present at the 3\_dimensional coordinate which is present there. Time complexity is O(n).

Delete high data – It is used to free the memory of the node present at the 3\_dimensional coordinates. Time complexity is O(n).

# LOCATING ROUTES [VERSION 3]



Version 3 is a different application of 3-D Linked list from version 1.

Here we have used the grid for computing the time to reach a location form one point to another. This project can be applied to those areas where the paths are in the form of a grid.

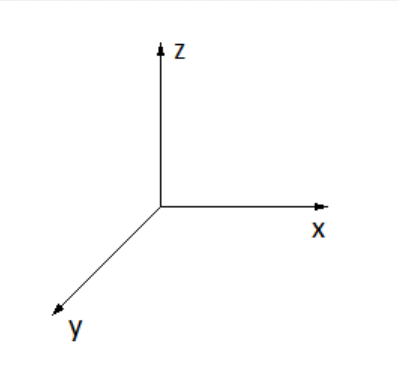
In the first level we have assigned the values of street name to the nodes in the intersection.

In the higher level we have put the values to reach the given node from the length.

In the further high level, we have placed the values of time to reach the given node from the breadth.

We have assumed that the time in to-and-fro between any 2 points is equal.

Direction Conventions are shown in the image below



So, if we are calculating the time going from one point to another along the length by following direction conventions then the time displayed over the high data of the node along the length is to be added and if we are going opposite to the direction convention then we have to add the data opposite to the node of where we are going.

Similarly, if we are calculating the time going from one point to another along the breadth by following direction conventions then the time displayed over the high data of the high data of the node along the breadth is to be added and if we are going opposite to the direction convention then we have to add the data opposite to the node of where we are going.

# SYNOPIS OF FUNCTIONS USED IN VERSION 3

As we have used singly linked list in joining 2 nodes, direction conventions are made to find the proper time in reaching the direction.

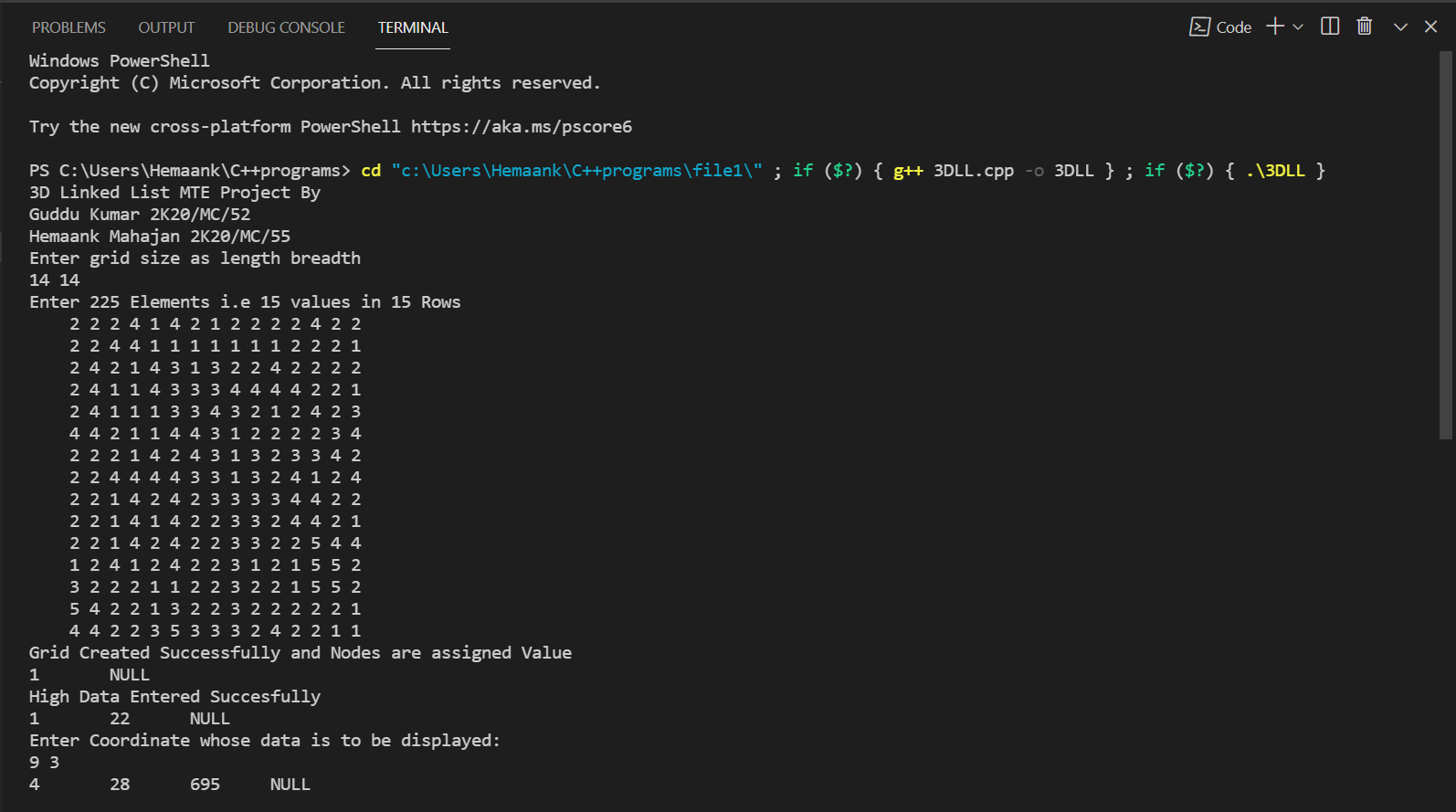
Give Node function – This function will return the address of the required node in the grid by passing the coordinates of x and y. Time complexity is O(n)

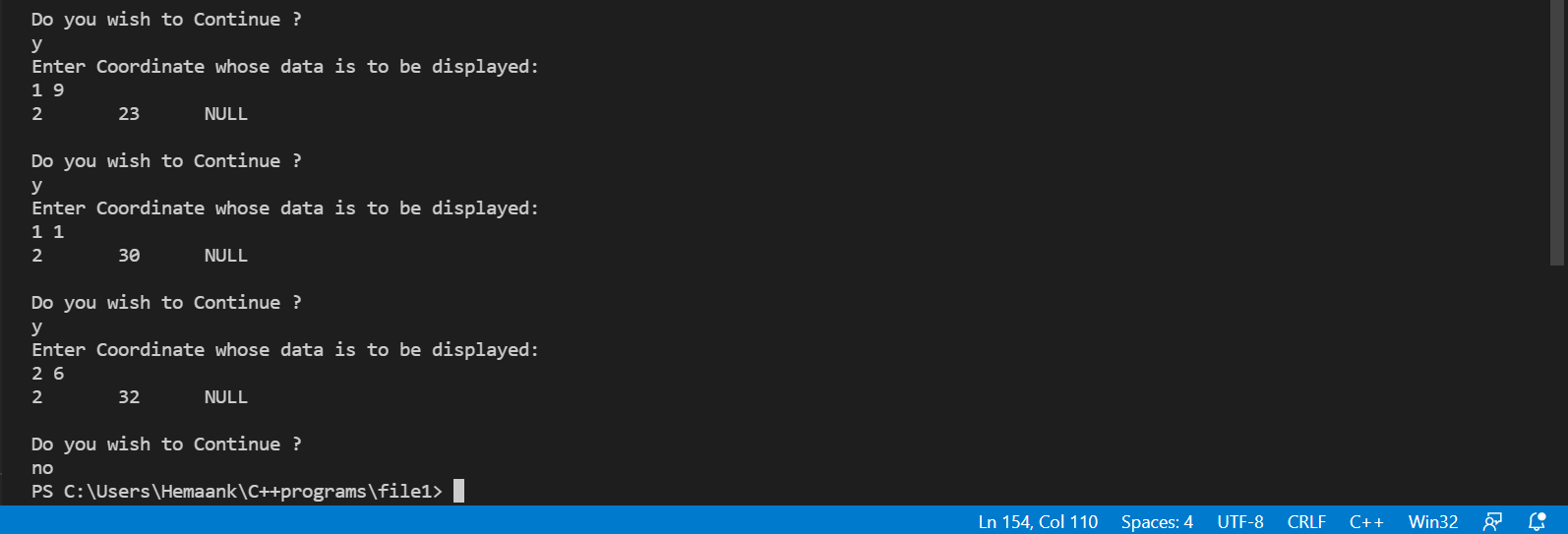
Route function – This function takes the coordinates of starting location/current location and destination location. As we cannot compare sufficiently large number of possible paths between 2 points, it will simultaneously cover 2 paths i.e., first along the length then breadth with first along the breadth then length and then compare them to return the path of smaller time interval and also prints that path in the output screen.

Here, autoroute function is a future work in the project where path of shortest time interval can be obtained mathematically.

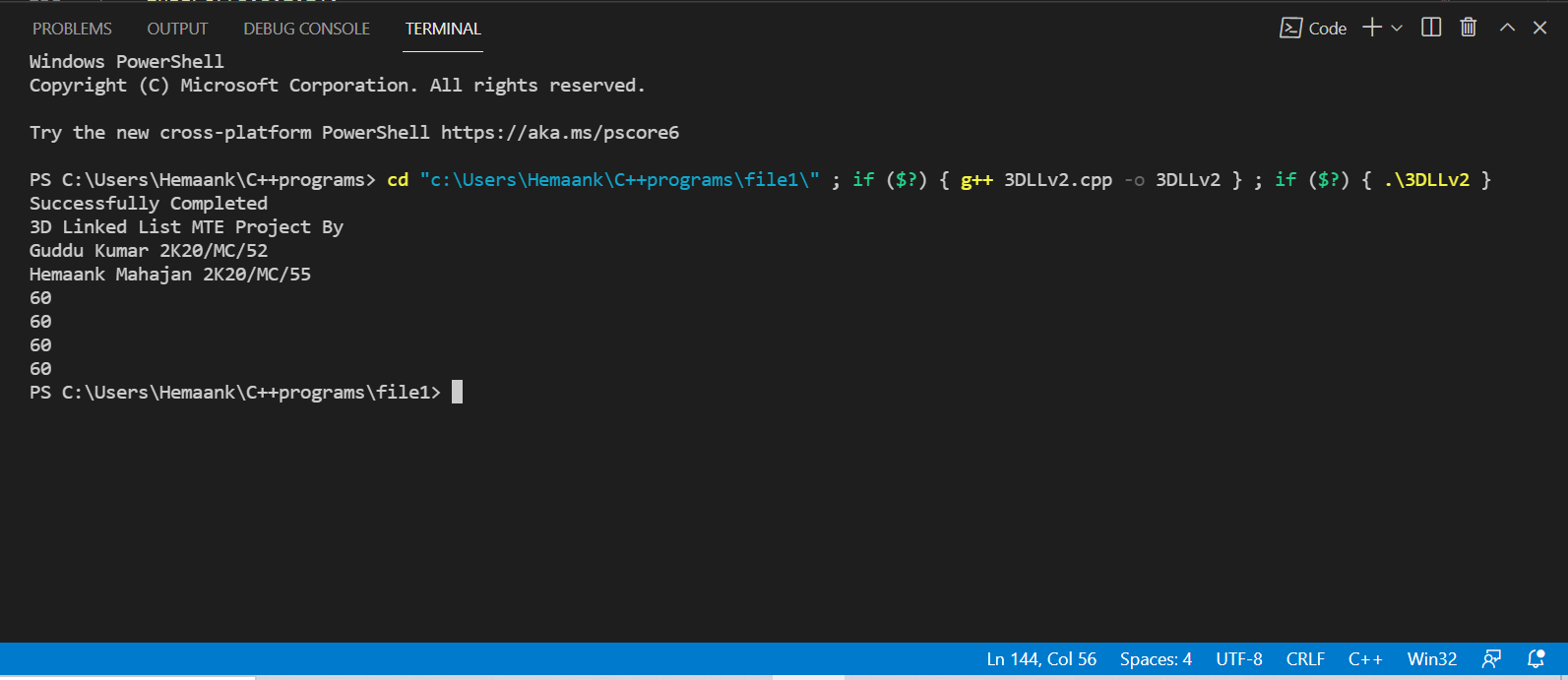
# OUTPUT IMAGES

## FOR VERSION 1



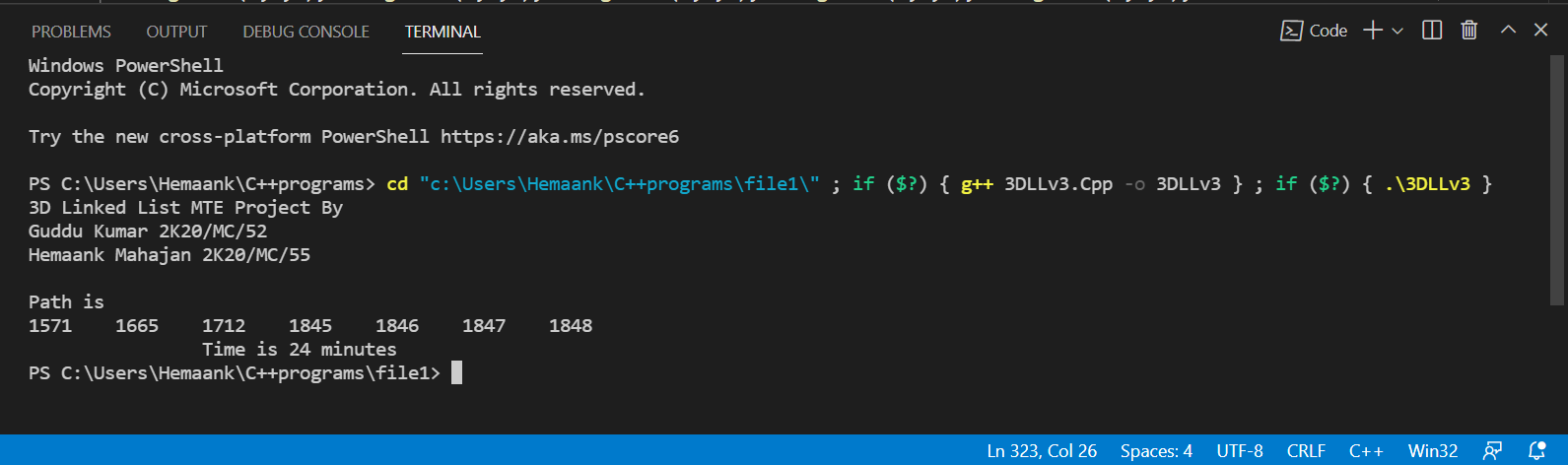


## FOR VERSION 2

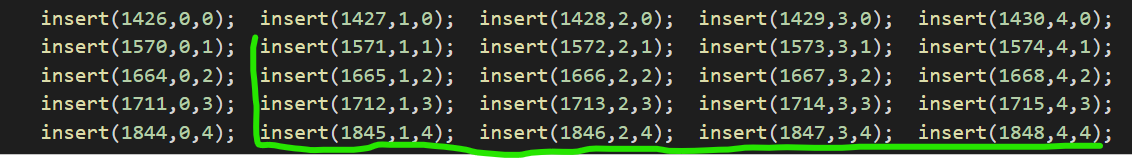


## FOR VERSION 3

Entered Coordinates are 1,1 to 4,4.



Which is interpreted as



# CONCLUSION

Hence, we have successfully executed all the versions and discussed about the general theory on them. So, 3-D Linked List turns out to be a useful data storing structure that can store data efficiently and provides flexibility in accessing the data. Its applications are only restricted to imagination. Here, we have taken 3 main applications to it that is mapping a city, forming a cuboid lattice and finding routes in a city. All the applications are also coded and presented herein. We have also provided many possible future works in these applications.

# REFERENCES

[1] INTRODUCTION TO ALGORITHMS THIRD EDITION – THOMAS H. CORMEN, CHARLES E. LEISERSON, RONALD L. RIVEST, CLIFFORD STEIN. The MIT Press Cambridge, Massachusetts London, England.

[2] DATA STRUCTURE AND ALGORITHMS MADE EASY -DATA STRUCTURE AND ALGORITHMS PUZZLES – 5TH EDITION – NARASIMHA KARUMANCHI. 2017 CAREERMONK Publications