Name: Kushagra Gupta

Student Id: 804729

Part 6

If we were provided with an adjacency matrix instead of an adjacency list then:

a) Depth-first traversal: We would just need an array to keep track of the visited nodes and would have gone through each cell of the adjacency matrix and check for If the node was visited or not. If the node was not visited then we could have printed it and if it was we could just have continued moving through the adjacency matrix.
b) Breadth-first traversal: We would just need an array to keep track of the visited nodes and would have gone through each cell of the adjacency matrix by looping row-wise through the adjacency matrix and check for If the node was visited or not. If the node was not visited then we could have printed it and if it was we could just have continued moving through the adjacency matrix.

Part 7

3) The detailed_path function makes use of the depth-first-search algorithm to traverse through the graph from the source to the destination and keeps on printing the places visited in order to get from the source and the node. How is basically works is that it goes on till there is something on the stack and starts with placing the source on the stack then goes to one of its adjacent nodes and checks if it is the destination prints and breaks free from the loop or else checks if the node is not visited, if not then it pushes that onto to the stack and does this kind of recursive thing till you find a NULL and then pops one thing out of the stack and prints it.

4)the all_paths functions uses a helper print function which has an array “path” passed to it that keep tracks of the visited path in a particular call and also and array “visited” passed to it that note of the visited nodes. The function checks that if the destination is reached from a particular source then it prints out the path using the “path” array; if the destination is not reached it traverse through rather recursively changing the source to an adjacent node of the previous node.

5)the function shortest_path uses the same helper function as it is going through all the paths to find the shortest among them.The helper function uses an array to keep track of the shortest path and also a variable which contains the shortest part. The function checks that if the destination is reached from a particular source then it checks if the sum of the

weights of all the nodes that are in the path against the minimum sum for far and updates the minimum accordingly; if the destination is not reached it traverse through rather recursively changing the source to an adjacent node of the previous node, also adding up the weights as it recurses.