<u>Assessed Coursework MA314, LT 2019, Candidate Number:14459</u>

a) Describe an efficient algorithm for determining whether a graph is bipartite.

We can use Breadth-First Search(BFS) to attempt a partition of vertices of a given graph into 2 sets of L and R. We can use colours to represent these 2 sets of L and R, White for L and Black for R. The algorithm would work as follows:

- Pick a vertex with no colour to begin and set its colour to White (assigned to set L)
- Set the colour of the adjacent vertices to Black (assigning them to set R)
- Set the colour of the vertices adjacent to the Black vertices to White (assigning them to set L)
- Repeat steps 2 and 3 until all vertices are coloured, if the graph is not strongly connected repeat step 1 in addition to the others
- When assigning colours, if the colour of an adjacent vertex is the same as the colour of the current vertex, then this implies that we can't assign all vertices to L and R without violating the bipartite condition. In this scenario we can deduce that the graph is not bipartite.

c) What is the running time of your algorithm in big O-Notation?

$O(n + m)$ *where n is the number of vertices and m is the number of edges in the graph*

d) Argue why your algorithm is correct

My algorithm uses BFS to visit each vertex and assigns it to either set L or R by changing its colour. For the initialization, we set the colour values of all vertices to None, just like the initialization of BFS without the distances. For maintenance, the algorithm uses BFS to visit each vertex, initially setting the starting vertex to White (if the graph is not strongly connected, then after all the connected vertices are coloured, this step is repeated for all the non connected parts of the graph), changing the colour values of all adjacent vertices that don't have an assigned colour to Black/White depending on the colour of the predecessor. The algorithm terminates under 3 conditions:

1. A self-loop
2. The colour of the predecessor is the same as the colour of the adjacent vertex
3. All vertices are coloured

If there is a self-loop, then the graph can never be bipartite as the self-loop vertex will always point to a vertex of the same colour(itself). Hence the algorithm returns false under this condition. Condition 2 implies that the vertices can not be divided into 2 sets without violating the bipartite condition, hence implying that the graph is not bipartite and returning False. Condition 3 implies that all vertices have been visited and both condition 1 and 2 are false,implying that the graphs vertices can be divided into 2 sets. Therefore we can deduce that the graph is bipartite and our algorithm returns True. From this we can see that our algorithm works as intended, therefore this algorithm is correct.