Q1)

1) Depth First Search = R5
   It did not produce an optimal path because depth first search is an uninformed search which explores nodes blindly until maximum depth is reached and if it does not find the goal node then it expands the children of adjacent child nodes similarly. Hence, the path found is longer than the actual one.

2) Breadth First Search = R1
   Again, this search is also an uninformed and blind search therefore it may explore more nodes than required to reach the goal state.

3) Uniform Cost Search = R7
   It will produce the path with cheapest cost so its result can be considered optimal.

4) Best First Search (H1) = R2
   It is not optimal because it only considers the heuristic cost, so it expands wherever the heuristic cost is less regardless of the path cost therefore, a shorter path exists which is from C to G.

5) Best First Search (H2) = R4
   Although the path is optimal but that's not always the case in best first searches

6) A* search (H1) = R3
   Although A* search finds the optimal path but in this case since the heuristic costs are not consistent and are a bad approximation therefore, the solution find is not the shortest one.

7) A* search (H2) = R6
   In this case, the heuristic costs were consistent and a good approximation therefore, the A* search in this case found the most optimal path.

Q2)

1) Depth First Search = A, B, E, F, G, C, H, D, I, Z
2) Best First Search = A, B, E, F, G, D, I, Z
3) A* search = A, B, D, Z
4) Breadth First Search = A, B, C, D, E, F, G, H, I, Z

Q3) i) In Depth-First search Stack data structure is used while in Breadth First Search Queue Data Structure is used. DFS is used to explore the deepest nodes first by expanding all the children of a node first before expanding the adjacent nodes. It is suitable in situations like puzzles or game problems where all the paths generating from a decision are explored to check if it leads to the win situation. Whereas, in BFS shallowest nodes are explored first by expanding all the adjacent nodes and then expanding their children, respectively. It is suitable for situations where goal is nearer to the starting node unlike games or puzzles problem.

ii) There is a problem with depth first search that if a node has very large depth or even infinite depth then the search algorithm can get stuck searching through that depth for a very long or even infinite time. Hence, depth limited search is used to overcome this infinite depth problem where a restriction is put on the depth to be explored. For instance, if the depth limit is d, then all the nodes will be explored until the depth is d after that they will be treated as if they have no child nodes and hence, the algorithm will jump to the adjacent node.

iii) A* search is an informed search which can be used to find the shortest path towards the goal node. It uses F cost to decide which node to expand next. It expands nodes with lowest F cost until the goal node is reached. F cost of any node is basically the sum of its path cost and heuristic cost. The path cost is the total cost to traverse to that current node from the starting node while heuristic cost is the approximate cost of goal node from the current node. Heuristic cost can be approximated through different distances like Manhattan distance, diagonal distance, and Euclidean distance. All these distances are the distance from current node to the goal node.

iv) Both uniform cost search and A* search are informed search which expand nodes with lowest cost. However, they differ in the cost they use to expand nodes. Uniform cost search uses cumulative path cost from starting node to the current node while A* search uses F cost which is the sum of cumulative path cost and heuristic cost of the node. As a result, uniform cost search can expand in any direction while A* search only expands in directions which takes it nearer to the goal node.

iv) Breadth-First search, Uniform Cost search and A* search are all complete because they will ultimately find the goal node. Uniform Cost search and A* search both are also optimal whereas, BFS is only optimal when all the actions have same cost.