Introduction: Your beloved aunt Petunia is hosting her namesday party and has requested your assistance in setting up a seating scheme. The objective is to ensure that guests who dislike each other are not seated at the same table. To achieve this, we have developed a recursive Depth-First Search (DFS) algorithm.

Solution Description:

1. Graph Representation:
   * The guests and their animosities are represented using an unordered map, graph.
   * Each guest is a key in the map, and their value is a vector containing the names of guests they dislike.
2. Depth-First Search (DFS) Algorithm:
   * We use a recursive DFS algorithm to traverse the graph and form separate tables for guests who do not dislike each other.
   * The DFS algorithm starts with an unvisited guest and explores all their neighbors (guests they dislike).
   * It continues recursively until all connected guests are visited.
   * During the DFS traversal, we maintain a visited set to keep track of the visited guests and avoid revisiting them.
3. Setting up the Seating Scheme:
   * We implement the dfs function that performs the DFS traversal and assigns a table number to each guest.
   * The dfs function takes the graph, a guest, the seating arrangement map, and the visited set as parameters.
   * When a guest is visited, it is added to the visited set and assigned the current table number.
   * The seating\_arrangement map stores the seating arrangement, where each guest is associated with their respective table number.
4. Generating the Seating Scheme:
   * We define the setupSittingScheme function that initializes the seating arrangement and calls the DFS algorithm for unvisited guests.
   * The function iterates through all guests in the graph and assigns table numbers to unvisited guests using the dfs function.
   * The table number is incremented for each new connected component.
5. Printing the Seating Scheme:
   * In the main function, we create an instance of the NamesdayParty class and populate the graph with invited guests and their animosities.
   * We call the setSittingScheme function to obtain the seating arrangement as a map.
   * Finally, we print the seating arrangement by iterating over the map and displaying each guest's name and their associated table number.

Conclusion: The recursive DFS algorithm implemented in this solution provides an efficient way to set up a seating scheme for your aunt's namesday party. It ensures that guests who dislike each other are placed at separate tables, promoting a harmonious atmosphere during the celebration. By utilizing a graph representation and recursive traversal, the solution accommodates various animosities among the invited guests.