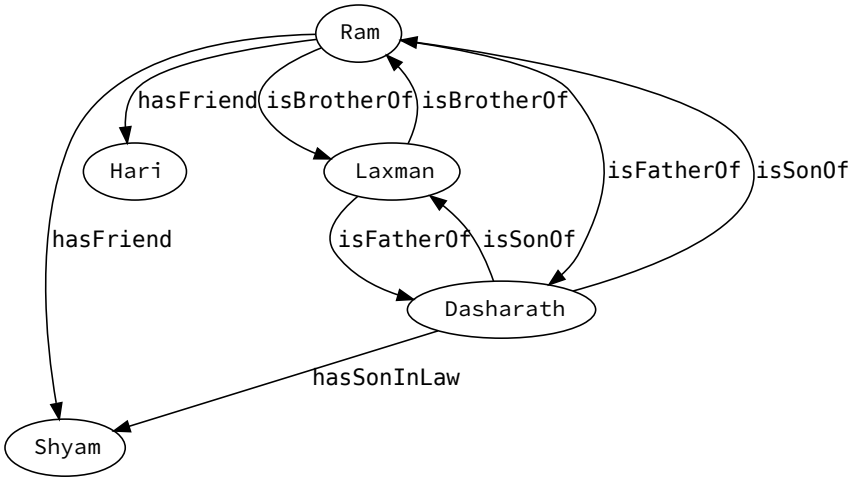
Directed Graph

Time Span: 2 lab days.

Problem: Write a C program to represent a family members and their relations with a directed graph.

Description:
The following diagram describes a typical family members and their available relationships.



Family Relations Representation in a Graph.

In the above graph, there exists multiple relations between two vertices. For example, Shyam has friend relations with Ram and His father Dasharath has son in law relation. You are required to implement all the given relations in a directed graph with **adjacency lists** or with an **adjacency matrix** through C/C++ language. Implement the following function modules mentioned in tasks section.

Tasks:

1. **initFamilyGraph()** to perform initialization of the graph in memory. During initialization, you need to create and add all the vertices and edges in the family graph.

2. **isVertexPresent(vertexName)** to search the vertex name through the given list of vertices.

   Example: isVertexPresent(“Dasharath”) should return 1, but isVertexPresent(“Samuel”) should return 0.

3. **isRelationExists(sourceName, relationName, targetName)** to find if the relation name available in between the source and the target name of the vertices.

   Example: isRelationExists(“Ram”, “isSonOf”, “Dasharath”) should return 1, but isRelationExists(“Ram”, “isFatherOf”, “Dasharath”) should return 0.

4. **displayVertexRelations(vertexName)** to display all the available relations in the given vertex name.

   Example: displayVertexRelations(“Dasharath”) should return a list of relations with [hasSonInLaw, isFatherOf, isFatherOf].

5. **displayPathsOf(sourceName, targetName)** to display all the available paths between the given vertices.

   Example: Available paths from “Ram” to “Shyam” are:
   1. hasFriend
   2. isFatherOf, hasSonInLaw

***