**Assignment 3**

**Implement a Knowledge Base for Family Relationships**

**Theory -**

A Knowledge Base (KB) is a technology used to store complex structured and unstructured information used by a computer system. The purpose is to make information explicit and to reason about it automatically.

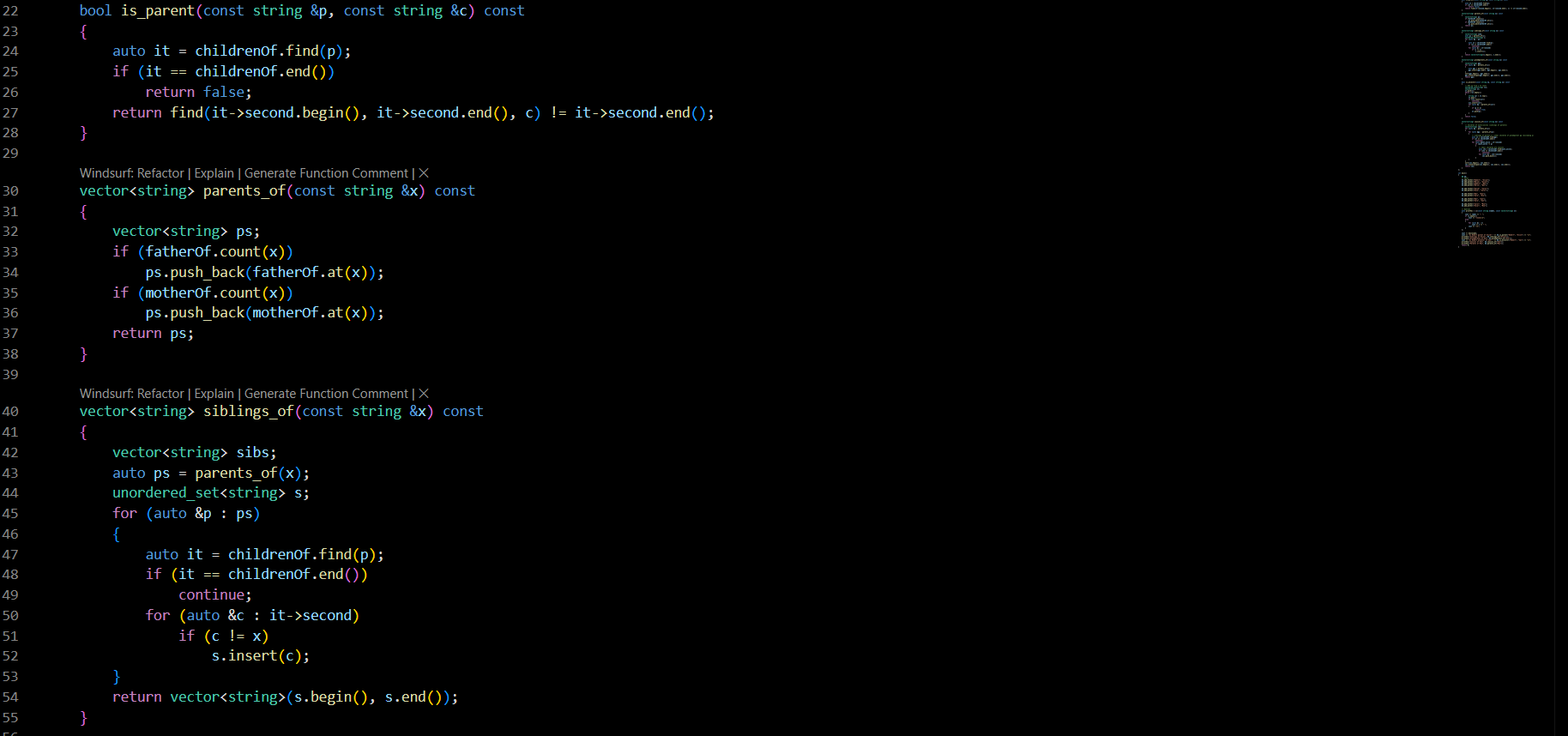
This implementation creates a simple knowledge base to represent family relationships. It operates on two fundamental principles:

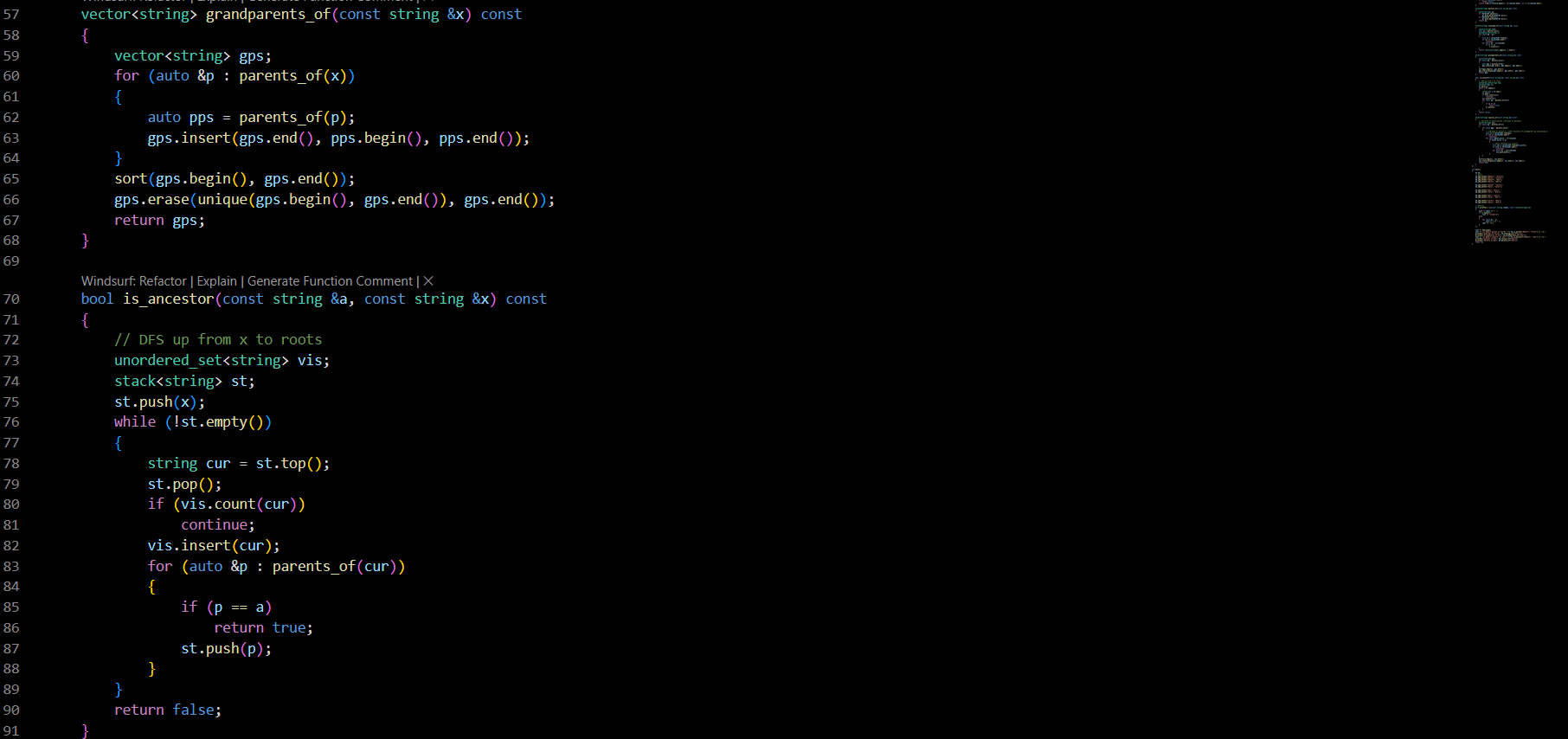
* Facts: These are the basic, explicitly stated pieces of information that are added to the KB. In this case, the facts are parent-child relationships, specifically father(X, Y) and mother(X, Y), meaning X is the father/mother of Y.
* Queries (Derived Relations): These are logical rules or functions that infer more complex relationships from the basic facts. The system can answer queries about relationships that are not explicitly stored, such as identifying siblings, grandparents, ancestors, and cousins.

The goal is to build a system that, given a set of simple parent-child facts, can intelligently answer queries about the broader family tree structure.

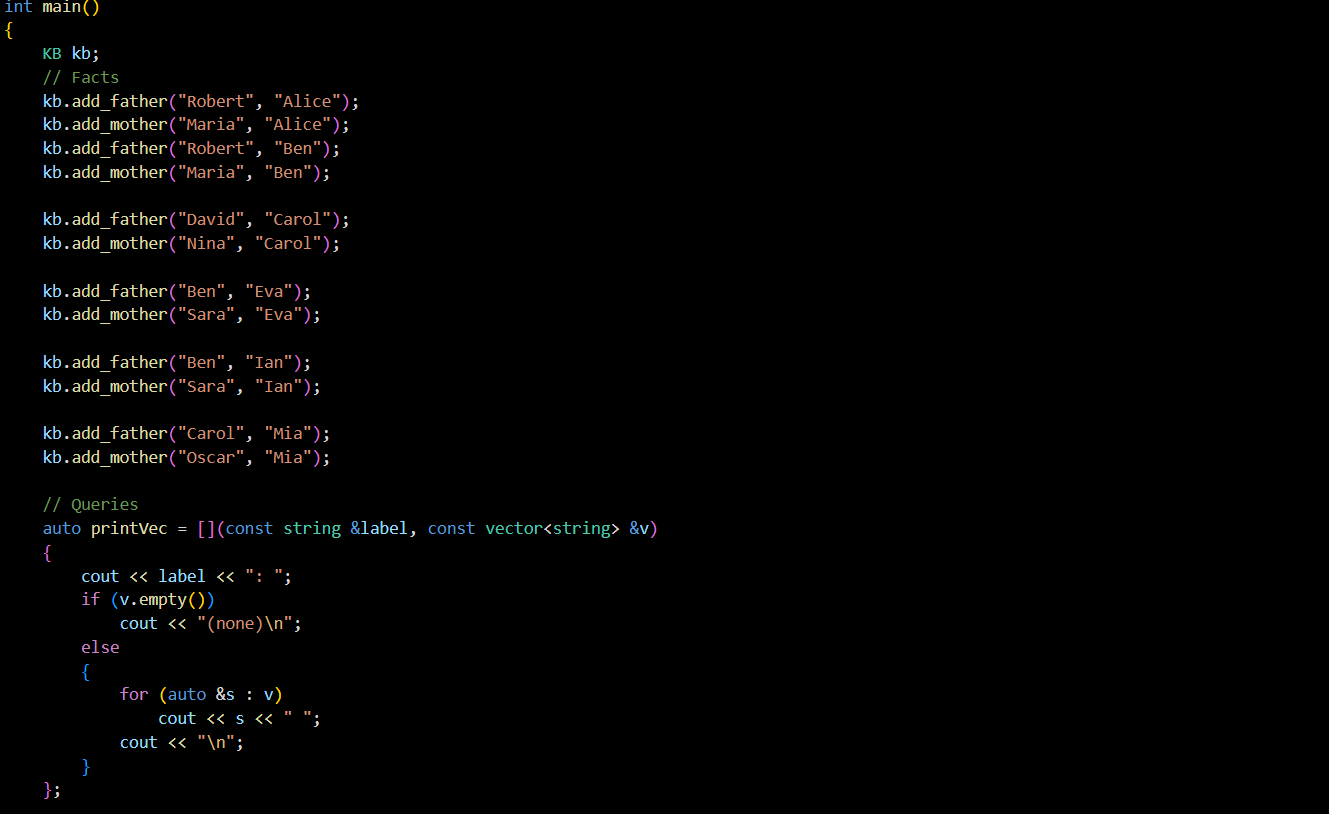
**Code –**

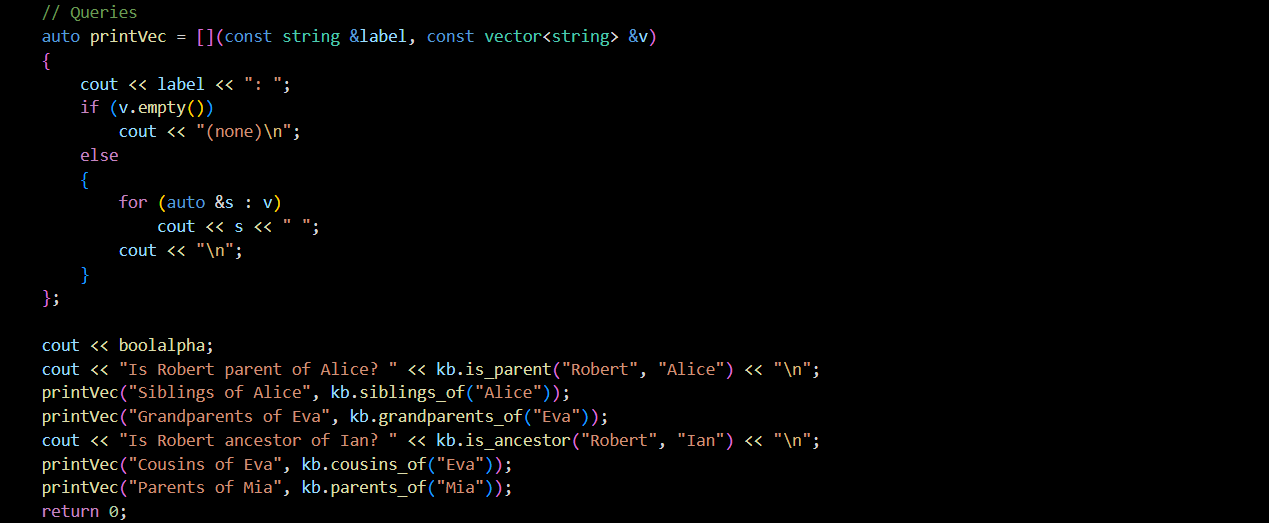
****

****

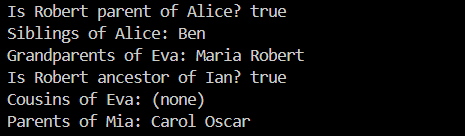
****

****

****

****

**Output –**

****

**Explanation –**

Representation of Knowledge: The family tree is stored in three hash maps (unordered\_map) for efficient lookups. fatherOf and motherOf map a child to their father and mother, respectively. childrenOf maps a parent to a list of all their children.

Adding Facts: The add\_father and add\_mother functions are used to populate the knowledge base. Each call adds two pieces of information: the direct parent-child link (e.g., fatherOf["Alice"] = "Robert") and the reverse child-parent link (e.g., childrenOf["Robert"] gets "Alice" added to its list).

Querying and Reasoning:

* parents\_of(X): This is the simplest query. It checks the fatherOf and motherOf maps to find the direct parents of X.
* siblings\_of(X): This function first finds the parents of X. Then, for each parent, it retrieves their list of children and adds any child who is not X to the result.
* grandparents\_of(X): This is a two-step query. It first finds the parents of X, and then for each of those parents, it calls parents\_of again to find their parents.
* is\_ancestor(A, X): This function determines if person A is an ancestor of person X. It performs a Depth-First Search (DFS) starting from X and moving upwards through the family tree. It uses a stack to keep track of parents, grandparents, etc., to check if A is ever encountered.
* cousins\_of(X): This derives the most complex relationship. It finds the parents of X, then their parents (the grandparents). For each grandparent, it finds their children (the aunts and uncles of X) and then finds the children of those aunts and uncles, who are the cousins of X.

Execution: The main function initializes the KB object, populates it with a series of add\_father and add\_mother facts, and then runs several queries to demonstrate that the system can correctly infer relationships like siblings, grandparents, and ancestors.

**Conclusion -** This program successfully models a simple knowledge base for family relationships. By storing basic parent-child facts, it uses logical functions to reason about and derive more complex relations, demonstrating a fundamental concept in knowledge representation and artificial intelligence.