Prelab 10

In this prelab, you will implement a system to manage records using a Binary Search Tree (BST). Instead of working with an existing set of data, you'll be creating a dynamic system where records can be inserted and queried as needed.

The goal is to implement the following functions:

- **`RecordDB initializeRecordDB();`** `// Allocates and initializes your BST-based DB`
- **`RecordDB addRecord(Record *, RecordDB);`** `// Inserts a new record into the BST`
- **`int countRecordsInRange(int min, int max, RecordDB);`** `// Counts how many records fall within the specified range`
- **`void freeRecordDB(RecordDB);`** `// Frees memory allocated for the database`
- **`int getDatabaseErrorCode(RecordDB);`** `// Returns any error code from the database operations`

Key Details:

1. **Note:** The definitions the **`Record`** and **`RecordDB`** structures are up to you.
2. **addRecord**: This function will use the standard BST insertion technique, so it won't guarantee balanced tree performance (O(log N) complexity is not assured). You are responsible for implementing the insert logic for BST.
3. **initializeRecordDB**: This function initializes and allocates memory for the database (BST). It should be the starting point for using the database.
4. **countRecordsInRange**: This function will iterate over the BST and count how many records fall between a given minimum and maximum value.
5. **freeRecordDB**: Responsible for cleaning up and freeing all the memory allocated for the database and its records.
6. **getDatabaseErrorCode**: This function will help track if any errors occur during the operations (like insertion failure or memory allocation issues).