API Specifications

*** DATABASE:**

Database is a class. It has 7 functions as described below.

1. **Constructor ( )**
   - Purpose: Default constructor. Creates an empty database
   - Arguments: none
   - Return values: none
2. **Destructor ( )**
   - Purpose: Destroys the database object
   - Arguments: none
   - Return values: none
3. **void addTable (Table& table, const std::string& tableName)**
   - Purpose: Adds a single table to the database
   - Arguments: a table object, a string represents the table name
   - Return values: none (void)
4. **void dropTable (const std::string& tableName)**
   - Purpose: Deletes a table from the database
   - Argument: a string represents the table name
   - Return values: none (void)
5. **std::vector<std::string> listTables();**
   - Purpose: Lists all table names in the database
   - Arguments: none
   - Return value: a vector of all Table names
6. **std::vector<Table*> getTables();**
   - Purpose: Gets all tables in the database
   - Arguments: none
   - Return value: a vector of Table* objects
7. **Table Querry(std::string Select, std::string From, std::string Where);**
   - Purpose: Searches for a table from the database
   - Arguments:
     - + SELECT: either a list (in order) of attributes name or '*' for all attributes
     - + FROM: a single table name
     - + WHERE: references to the attribute name
   - Return value: a table from the database. If a table not found, returns a warning statement

*** TABLE:**

Table is a class with 15 functions.

1. **Constructor ( )**
   Purpose: Default constructor. Creates an empty table
   Arguments: none
   Return values: none
2. **Table(std::vector<std::string> attributeNames)**
   Purpose: Creates a table with given attribute names
   Arguments: a vector of all attribute names.
   Return values: none (void)
3. **Table(const Table& t)**
   Purpose: Copy constructor. Allows table to be copied
   Arguments: none
   Return values: none
4. **~Table ( )**
   Purpose: Destructor. Destroys the table object
   Arguments: none
   Return values: none
5. **std::string getTableKey() const**
   Purpose: Gets the table Key
   Arguments: none
   Return value: a string represents the table key
6. **void specifyKey(const std::string& key)**
   Purpose: Allows an attribute name to be the key of the table
   Argument: a string represents the attribute name
   Return values: none (void)
7. **void addAttribute(std::string& attributeName)**
   Purpose: Add a new column to the end of the table with the attribute name
   Note: All entries in the newly created column should initially be NULL
   Argument: an attribute name
   Return values: none (void)
8. **void deleteAttribute(std::string& attributeName)**
   Purpose: Deletes a column with the given attribute name from the table
   Argument: an attribute name
   Return values: none (void)
9. **void insertRecord(Record& record);**
   Purpose: Adds a record to the table
   Argument: a record reference
   Return values: none (void)

10. **std::vector<std::string> getAttributeNames( )**
    Purpose: Returns an ordered list of attribute names for the table
    Arguments: none
    Return value: an ordered list of attribute names
11. **int getTableSize( );**
    Purpose: Returns the number of records in the table
    Arguments: none
    Return value: number of records in the table
12. **Table crossJoin(const Table* table1, const Table* table2)**
    Purpose: Merges two tables
    Arguments: 2 tables that need to be merged
    Return value: a merged table
13. **Table naturalJoin(const Table* table1, const Table* table2)**
    Purpose: Merges two tables
    Note:
        + First table should have attribute name that matches the key from the 2nd table
        + Should create one entry for each row of the first table, with the additional columns from the matching key in the second table
        + An exception should be throw when the second table does not have a key, or the First table does not have an attribute matching the key name.
    Arguments: 2 tables that need to be merged
    Return value: a merged table
14. **std::tuple<std::string,std::string,std::string> findRoutine(const std::string& attributeName)**
    Purpose: count the number of elements, find max element, and find min element
    Argument: an attribute name
    Return value: a tuple of number of elements, maximum element, and minimum element
15. **Iterator ( )**
    Purpose: iterator for the containers in the table class. All iterators are private.

*** RECORD**

NOTE: ALL DATAS ARE OF TYPE STRING

1. **Record ( )**
    Purpose: Default constructor. Creates a record of arbitrary size and initialize all entries to NULL string
    Arguments: none
    Return values: none

2. **Record (const Record& record)**
   Purpose: Copy constructor. Avoids shallow copy of record object
   Argument: a record object
   Return values: none
3. **Record& Record::operator = (const Record& record)**
   Purpose: Assignment operator. Avoids shallow copy of record object
   Argument: a record object
   Return values: none
4. **~ Record ( )**
   Purpose: Destructor. Destroys a record object.
   Arguments: none
   Return values: none
5. **int getRecordSize ( )**
   Purpose: Finds the number of elements in a record
   Arguments: none
   Return value: number of elements in the record
6. **std::string& operator [] (int index)**
   Purpose: Accessor operator. Allows individual entries to be read/written
   Arguments: none
   Return value: a value at the desired position