Project A:
Database Data Structure

Directions:

A database is a collection of tables that consists of records called entities (or tuples). An entity consists of attributes which are properties (or fields) of the entity. Each attribute has a domain which is a set of permitted values; it can be thought of as the data type of the attribute. Moreover, a table only consist of entities with the same attribute format. Tables allows insertions, removals and queries (conditional searches). However, an entity can only be added to a table if it has a key, which is an attribute or set of attributes used to identify an entity, that is distinct from the other entities already stored in the table. For instance, the following table consists of entities with the attributes *first name* (string domain), *last name* (string domain), and *email* (email domain). And its key would be $\{$*first name*, *last name*$\}$

Email Contacts

First Name	Last Name	Email
John	Doe	johndoe@mec.science
Jane	Doe	janedoe@mec.science
Mary	Jane	maryjane@mec.science
Mary	Parker	maryparker@mec.science
Peter	Parker	peterparker@mec.science

where an email domain a string domain with a specific format.

For your project, you will be creating a database consisting of one table. Your project will consist of the following properties and constraints

- □ A generic interface named **Domain** that contains
  - a constant generic type reference constant get method that takes no parameters.
  - a void set method that takes a constant generic type reference parameter.
  - a bool constant method named `Equals()` that takes a constant generic **Domain** reference parameter.
  - a string constant method named `ToString()` that takes no parameters.
- □ A derived **Domain** class for each distinct domain type for the attributes of your specified table that
  - defines all of its special member functions using the specified properties.
  - defines all the methods of the interface **Domain** using the specified constraints.
  - defines an overloaded ostream operator. It displays the same format as `ToString()`.
  - defines an overloaded equal operator (`==`) that takes two constant derived class reference parameters.
- □ A class named **Entity** that
  - contains public field for each of its attributes.
  - defines all of its special member functions.
  - defines a string constant method named `ToString()` that takes no parameters. It creates a string that is a list of the attributes each separated by a comma all enclosed in parentheses.
  - defines an overloaded ostream operator. It displays the same format as `ToString()`.
  - defines an overloaded equal operator (`==`) that takes two constant **Entity** reference parameters. It returns true only if the keys of the parameters are equal.
- □ A class named **Table** that
  - contains a private **Entity Array** field of a size of 500.
  - contains a private ulong field that represents the amount of entities in the table.

- defines all of its special member functions.
- defines a string constant method named `ToString()` that takes no parameters. It creates a string that is a list of its entites each on their own line.
- defines a void method named `Insert()` that takes a constant **Entity** reference parameter. It adds the parameter to the table if the table is not full and the key of the parameter is not in the table.
- defines a void method named `Remove()` that takes a constant **Entity** reference parameter. It removes the entity from the table whose key matches the key of the parameter.
- defines an ulong constant method named `Count()` that takes no parameters. It returns the amount of entities in the table.
- defines an overloaded ostream operator. It displays the same format as `ToString()`.
- defines a constant **Entity** reference constant overloaded subscript operator that takes an ulong parameter. If the parameter represents a valid index, it returns a element of the table whose index is equal to the parameter; otherwise, it throws an error.

□ Each of the classes must be written in their own header file.

□ The header files can only include the libraries *iostream*, *string*, *sstream*, *cstdlib*, *cctype* and *Array.h*.