NTUST OOP Midterm Problem Design

Subject：Library Database

Author：杜竑毅

Main testing concept：STRUCT AND CLASS

Basics	Functions
■ C++ BASICS	□ SEPARATE COMPILATION AND NAMESPACES
■ FLOW OF CONTROL	□ STREAMS AND FILE I/O
□ FUNCTION BASICS	□ RECURSION
□ PARAMETERS AND OVERLOADING	□ INHERITANCE
□ ARRAYS	□ POLYMORPHISM AND VIRTUAL FUNCTIONS
■ STRUCTURES AND CLASSES	□ TEMPLATES
□ CONSTRUCTORS AND OTHER TOOLS	□ LINKED DATA STRUCTURES
□ OPERATOR OVERLOADING, FRIENDS,AND REFERENCES	□ EXCEPTION HANDLING
	■ STANDARD TEMPLATE LIBRARY
□ STRINGS	□ PATTERNS AND UML
□ POINTERS AND DYNAMIC ARRAYS	

Description：

A database is an organized collection of structured information, or data, typically stored electronically in a computer system, your job is to implement a library database system to store the information of books while providing simple commands to manage the database

The database system has 2 storage units: Collect and Book,

each Collect contains:

1. Its name.
2. Those books stored in it.

each Book contains:

1. bookId is an arbitrary but unique number.
2. bookName is an arbitrary string
3. 13-digit ISBN is a numeric commercial book identifier that is intended to be unique, and the first 12 digits record the registration and publication information, and the last digit is a check digit to indicate the validity of the ISBN code. Its validation method is listed in Notes at the end.

Input：

You will need to implement the following commands to manage the database, and there are totally seven different commands. Generally, a user issues a command in a line.

The operation corresponding to the commands in following:

- **Make:**

  Command: Make <collectName>

  To insert a collect takes one string, <**collectName>,** as input, adds an empty collect into the database, and outputs a message of "*Insert Collect: **collectName**.\n*" if it does

not exist. Otherwise, you should output a message "*Collect already exist\n*".

- **Drop:**
  Command: Drop <collectName>
  To delete a collect takes one string, <**collectName>,** as input, locates and removes the specific collect matching the <**collectName>** from the database, and outputs a message of "*Delete Collect:* ***collectName*** *with* ***N*** *books.\n*" if it exists. Otherwise, you should output a message of "*Collect doesn't exist.\n*".
- **Insert:**

  Command: Insert <collectName> <…books>

  To insert a set of books takes two strings, <**collectName>** and <…**books>,** as input, **<...books>** consists of "**<bookId>**,**<bookName>**,**<isbn>**", multiple books are linked using ',', adds **books** into the collect under the name of **collectName** and outputs the message of "*Insert* ***N*** *books into* ***collectName****.\n*", if all inputs are valid. Otherwise, you should detect the errors and output a message based on the following criteria:

  1. "*Collect doesn't exist.\n*": <**collectName>** does not exist in the database.
  2. "*Invalid bookid.\n*": <**bookId**> has already existed in the database or <**bookId**> is not a number.
  3. "*Invalid isbn.\n*": <**isbn**> has already existed in the database or <**isbn**> is not valid *(see other note).

- **Delete:**
  Command: Delete <…bookIds> <…isbn>
  To delete books takes one or two strings, <…**bookIds**> and <…**isbns**> as input, deletes all data matching the record of <…**bookIds**> and <…**isbns**> in the database, and outputs a message of "*N books delete.\n*"
- **Sort by bookId:**
  Command: Sort by bookId <collectName>
  To sort the collect based on the **bookId** takes <**collectName>** as input, sorts the collect in the descending order based on their **bookId**, and outputs the sorted records in the format of "*BookId:* ***bookId****\tBookName:* ***bookName****\tISBN:* ***isbn\n*** *<B1, B2, …>\n*", where B1, B2, … are their **bookId**, if <**collectName>** exists and is not empty. If <**collectName>** exists but is empty, you should output "*Collect is empty.\n*". Otherwise, you should output a "*Collect doesn't exist.\n*".
- **Sort by ISBN:**
  Command: Sort by ISBN <collectName>
  To sort the collect based on the **ISBN** takes <**collectName>** as input, sorts the collect in descending order based on their **ISBN**, and outputs the sorted records in the format of "*BookId:* ***bookId****\tBookName:* ***bookName****\tISBN:* ***isbn\n*** *<B1, B2, …>\n*", where B1, B2, … are the book records, if <**collectName>** exists and is not empty. If <**collectName>** exists but is empty, you should output "*Collect is empty.\n*". Otherwise, you should output a "*Collect doesn't exist.\n*".
- If the command doesn't exist, output the message of "*Unknown Command.\n*".
- If the command is not complete, output the message of "*Incomplete Command.\n*".

User can keep entering commands until reading EOF.

Output：

- The output message with corresponding input command.
- If the command doesn't exist, output the message of "*Unknown Command.\n*".
- If the command is not complete, output the message of "*Incomplete Command.\n*"

See the sample output.

Sample Input / Output :

Sample Input
Insert colle01 1,book1,9789866052675 Make colle01 Make colle01 Insert colle01 1,book1,9789866052675,2,book2,9789864762859,3,book3,9789865024864 Sort by bookId colle01 Delete 1,5,6 Sort by ISBN colle01 insert Insert 5
Sample Output
Collect doesn't exist. Insert Collect: colle01. Collect already exist. Insert 3 books into colle01. BookId: 1 BookName: book1 ISBN: 9789866052675 BookId: 2 BookName: book2 ISBN: 9789864762859 BookId: 3 BookName: book3 ISBN: 9789865024864 1 books delete. BookId: 2 BookName: book2 ISBN: 9789864762859 BookId: 3 BookName: book3 ISBN: 9789865024864 Unknown Command. Incomplete Command.

□ **Easy, only basic programming syntax and structure are required.**

□ **Medium, multiple programming grammars and structures are required.**

■ **Hard, need to use multiple program structures or complex data types.**

Expected solving time:

50 minutes

Other notes:

You are suggested to use STL such as std::tuple, std::pair, std::map, std::set and std::vector, to implement the database.

The calculation of an ISBN-13 check digit begins with the first twelve digits of the 13-digit ISBN (thus excluding the check digit itself). Each digit, from left to right, is alternately multiplied by 1 or 3, then those products are summed modulo 10 to give a value ranging from 0 to 9. Subtracted from 10, that leaves a result from 1 to 10. A zero replaces a ten(use X), so, in all cases, a single check digit results.

For example, the ISBN-13 check digit of 978-0-306-40615-*?* is calculated as follows:

$s = 9\times1 + 7\times3 + 8\times1 + 0\times3 + 3\times1 + 0\times3 + 6\times1 + 4\times3 + 0\times1 + 6\times3 + 1\times1 + 5\times3$

$= 9 + 21 + 8 + 0 + 3 + 0 + 6 + 12 + 0 + 18 + 1 + 15$

$= 93$

93 / 10 = 9 remainder 3
10 – 3 = 7