

## Assignment 6: Template and Linked List

### Creating an Information Repository with Various Data Types

#### Specification:

In this assignment, you will create different information repository databases that store information for various data types. Several information repositories can be created. A particular information can also be retrieved from an information repository. For example, a company repository database may look like the following:

345671	Austen, Jane	90000.00
564313	White, Anne	89000.00
783421	Mason, James	76000.00

Here, the first column represents the id (integer) of each employee, the second column represents the name (string) of each employee and the third column represents the annual salary (double) of each employee.

Similarly, a bookstore repository may contain the following information:

0-399-82477-1	The C Programming Language	30.00
0-201-88954-4	C++ Primer	40.00
0-201-82470-1	The Indispensable Guide to C	25.00

where, the first column represents the isbn (string) of each book, the second column represents the title (string) of each book and the third column represents the price (double) of each book.

A census repository database may look like the following:

201-84-4464	Foster, John	02-12-1975
659-43-259	Collins, Joan	01-12-1960
703-45-789	Adams, Jane	12-09-1971

Where, the first column represents the ssn (string) of each person, the second column represents the name and the third column shows the date of birth (string) of each person.

In each of these examples, the first column represents the key which will be used to retrieve an item (InfoNode) from a particular InfoRepository.

So, first of all, you need to create an information node that can hold multiple data types. So an InfoNode (information node) needs to be a Template class. An InfoNode can be described as a linked list which contains a pointer to the next InfoNode. An InfoNode class can be defined as follows:

```

template<class T, class U, class V>
class InfoRepository;

template<class T, class U, class V>
class InfoNode{
    friend class InfoRepository<T, U, V>;
public:
    InfoNode(T& t, U& u, V& v, InfoNode<T, U, V>* p) :dataOne(t), dataTwo(u), dataThree(v),
    next(p){}
private:
    T dataOne;
    U dataTwo;
    V dataThree;
    InfoNode* next; // points to next node in the list
};

```

As you can see, in **InfoNode** class, **InfoRepository** has been declared as a **friend** of **InfoNode** class. Because **InfoRepository** will store all the information for each **InfoNode** and needs to have access to the **InfoNodes** to retrieve or display information. Hence, **InfoRepository** class also needs to be declared as a **Template** class.

So **InfoRepository** class looks like the following:

```

template<class T, class U, class V>
class InfoRepository{
public:
    InfoRepository(std::string name) :first(0), iName(name){}
    ~InfoRepository();
    void addInfo(T t, U u, V v);
    bool isEmpty();
    void printInformation();
    void retrieveInfoNode(const T &t);

private:
    InfoNode<T, U, V>* first;
    std::string iName;
};

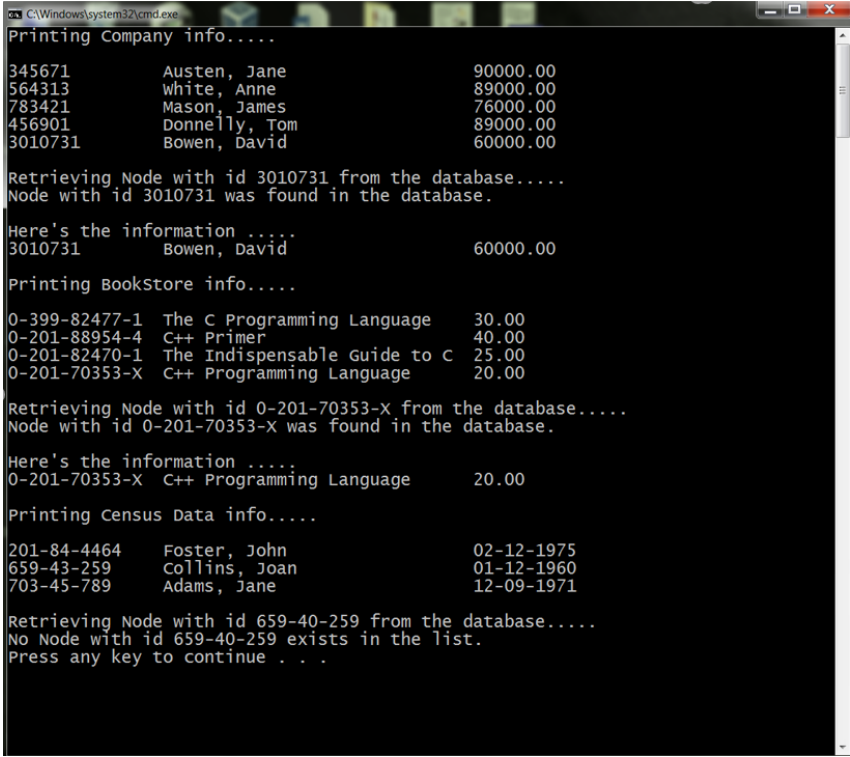
```

As already discussed, various **InfoRepository** containing different types of data can be created. Each **InfoRepository** object has its own name, i.e., “**Company Data**”, “**Book Store**”, “**Census Data**”, and so on. As **InfoNodes** are added to the **InfoRepository**, the last information is added to the top, represented as a pointer to the ‘**first**’ or the top **InfoNode** in the **InfoRepository** class. **InfoRepository** class can have several functions:  
**isEmpty()** method checks whether a particular **InfoRepository** is empty or not,  
**printInformation()** prints all the values of all **InfoNodes** in a particular **InfoRepository** database.

**retrieveInfoNode()** function takes the key as described above to search whether a particular **InfoNode** exists in an **InfoRepository**. If that particular **InfoNode** exists in the database, it retrieves and prints corresponding information. If that **InfoNode** does not exist, it just prints out that the corresponding **InfoNode** does not exist in the **InfoRepository** database.

So, you need to implement the functions as mentioned in the **InfoRepository** class. Use the attached tester file named **info\_repostory\_tester.cpp**. for the testing purpose.

The following output will be generated. Use “**iomanip**” library for proper formatting of the output as shown below.



```
C:\Windows\system32\cmd.exe
Printing Company info....
345671      Austen, Jane          90000.00
564313      White, Anne          89000.00
783421      Mason, James         76000.00
456901      Donnelly, Tom         89000.00
3010731     Bowen, David         60000.00

Retrieving Node with id 3010731 from the database.....
Node with id 3010731 was found in the database.

Here's the information .....
3010731     Bowen, David         60000.00

Printing BookStore info....
0-399-82477-1 The C Programming Language  30.00
0-201-88954-4 C++ Primer              40.00
0-201-82470-1 The Indispensable Guide to C  25.00
0-201-70353-X C++ Programming Language  20.00

Retrieving Node with id 0-201-70353-X from the database.....
Node with id 0-201-70353-X was found in the database.

Here's the information .....
0-201-70353-X C++ Programming Language  20.00

Printing Census Data info....
201-84-4464  Foster, John          02-12-1975
659-43-259   Collins, Joan         01-12-1960
703-45-789   Adams, Jane           12-09-1971

Retrieving Node with id 659-40-259 from the database.....
No Node with id 659-40-259 exists in the list.
Press any key to continue . . .
```

### Submission:

You need to complete the **InfoRepository** class by implementing all the functions as mentioned in the assignment specification. Compile and execute your code with g++ or the Microsoft compiler.

Place your solution in a zipped file named with your last name followed by the first initial of your first name followed by 6 (ex: **YasminS6.zip**) and submit the solution via canvas.

Submission deadline is Tuesday, **June 4, 11:59 pm**.

This assignment weighs **5%** of the course.