COSC 2320: Data Structures

Homework 2: Union and Intersection of Doubly Linked Lists using Recursion

Deadline: **Oct. 11, 2014**

1. INTRODUCTION

You will write a C++ program to find the intersection and/or union of two doubly linked lists using recursion. **You are not allowed to use the STL Library to implement linked list**.

A set is a collection of distinct entities regarded as a unit, being either individually specified or (more usually) satisfying specified conditions. In set theory the union of two sets A and B, is the set of all distinct elements in A and B. The intersection of two sets A and B is the set that contains all elements of A that also belong to B (or equivalently, all elements of B that also belong to A), but no other elements. In mathematics, the notion of bag (or multiset) is a generalization of the notion of set in which members are allowed to appear more than once. For this homework we will work with bags.

2. INPUT & OUTPUT

The input of the program is a text file with multiple commands to be executed by the program, one command per line. The commands are defined as follows:

- **Read** [*file name*] **[*list name*]**
  Read the specified file from disk and insert each word in the file into a doubly linked list named as **[*list name*]**. The file specified in this command, if exists, will be a text file, with an unbounded number of different words (case-sensitive) in **ALPHABETICAL ORDER**, separated by spaces, commas or periods. The input file can have several lines terminated by carriage returns. You can assume words will be at most **50** characters long and no special characters will appear in front or in the middle of each word.
  There can be arbitrary number of Read commands at any position of the command file. If two Read commands provided same list name, that list will be overwritten by the last Read command (all previous contents will be lost).
  Example:

  ```
  Read inputA.txt A
  Read inputB.txt B
  ```

  inputA.txt:

  ```
  Microsoft Google Facebook IBM
  Oracle Adobe Dell Lenovo
  ```

  inputB.txt:

  ```
  Dell Lenovo Sony Intel Samsung
  ```

- **Print [*list name*]** [*forward* | *backward*]
  Print all the elements in the linked list named **[*list name*]** to screen, one element per line. If there is no list having the given **[*list name*]**, skip the command. You should start from the head element if parameter “forward” is specified, from the tail element otherwise.

Example 1:

```
Print A forward
```

After executing the **Read** command above, the output of this command will be:

```
Microsoft
Google
Facebook
IBM
Oracle
Adobe
Dell
Lenovo
```

Example 2:

```
Print B backward
```

The output will be:

```
Samsung
Intel
Sony
Lenovo
Dell
```

- **Union** [*L1*] [*L2*] [*L3*]
  Find the union of sets L1 and L2 and store the result in L3.
  Example:

```
Union A B C
Print C forward
```

The output will be:

```
Microsoft
Google
Facebook
IBM
Oracle
Adobe
Dell
Lenovo
Sony
Intel
Samsung
```

- **Intersection** [*L1*] [*L2*] [*L3*]
  Find the intersection of sets L1 and L2 and store the result in L3.
  Example:

```
Intersection A B D
Print D forward
```

The output will be:

```
Dell
Lenovo
```

- **All undefined commands or commands with errors should be ignored.**

3. PROGRAM AND ARGUMENT SPECIFICATION

The main program should be called "**recursion**". The program should be able to take the first argument as input file name.

The call syntax will be like:

```
recursion.exe commands.txt
```

Note that the input file name will not necessarily be the same every time, so your program shouldn't have this "commands.txt" hard coded.

The elements and commands in this program are **CASE-SENSITIVE**. "Windows" and "windows" are considered as two different words!

Another important detail is that input files might be empty, since the empty set ($\emptyset$) is valid in set theory. You can use the properties of the empty set to produce your results:

$$A \cup \emptyset = A$$
$$A \cap \emptyset = \emptyset$$

Your program must be able to create and handle multiple doubly linked lists. (Hint: Think about creating a linked list of linked lists or an array of linked lists)

When performing the union and intersection of the linked lists **you must use ONLY recursion.** Iterators and loops are not allowed. This will be strictly enforced.

4. A SAMPLE TEST CASE

commands.txt:

```
Read inputA.txt A
Read inputB.txt B
Read inputC.txt C
Intersection A B D
Union A B E
Union D C F
Print E forward
Print F backward
```

inputA.txt:

```
Microsoft Google Facebook IBM
Oracle Adobe Dell Lenovo
```

inputB.txt:

```
Dell Lenovo Sony Intel Samsung
```

inputC.txt:

```

```

Program call:

```
recursion.exe commands.txt
```

Output:

```
Microsoft
Google
Facebook
IBM
Oracle
Adobe
Dell
Lenovo
Sony
Intel
Samsung

Lenovo
Dell
```

5. SUBMISSION REQUIREMENTS

Your submission should be well tested before submitting under Visual Studio 2010 or later versions. You can get a copy of Visual Studio from the UH website using your cougarnet username and password. The URL is http://uh.edu/infotech/php/software/list.php.

We use the UH blackboard system to collect your homework submissions. Before you submit your homework, please make sure to **put everything in a ZIP file** named in the form of **LastName_PeopleSoftID_HW2.zip.**

For example: Zhang_1234567_HW2.zip

The instructions about how to use the blackboard system can be found on the TA's webpage for this course: http://www2.cs.uh.edu/~yzhang/cosc2320-f2014/

6. GRADING

The maximum grade for this homework is 100.

You will get 15 pts for submitting the homework in time, 10 pts if your program can be successfully compiled.

We will test your program with 5 easy test cases and 5 hard test cases. Each easy test case will worth 10 pts, and each of the hard ones will worth 5 pts.

When testing, we will compare your program's output with the standard output. Therefore **do not print any content on the screen unless required**, avoid any prompt information like "Please enter the input file name:", "The elements in the doubly linked list are:" etc.

Last but not least, **no cheating or plagiarism will be tolerated in any graded submissions.**

If you have any other questions, please send email to zhangyiqun9164@gmail.com.