COSE213 Data Structures (Spring 2022)
Instructor: Prof. Won-Ki Jeong
Due date: April 17, 2022, 11:59 pm.

Assignment 2: Simple Text Editor

In this assignment, you will implement a doubly linked list using C++ template, and a simple line-based text editor using the implemented linked list data structure. Two main classes you need to implement are `DoublyLinkedList` and `TextEditor`.

1. Doubly linked list (50 pts)

Doubly linked list is a linked data structure that has two link fields, left link and right link, so that traversing the list can be done in both directions. You must implement the doubly linked list class that satisfies the following requirements:

- Node data type is defined by the template parameter.
- Copy constructor and assignment operator are implemented so that a new linked list can be created by an existing list.
- An empty list must contain a header node, and `first` pointer points to it.
- Assume that the traverse order is from left to right. Reverse traversal order is from right to left.
- The class should provide iterators for forward-order and reverse-order traversal of the list. You must provide start/end location of forward iterator using Begin()/End(), and those of reverse iterator using rBegin()/rEnd().
- ++ or – operator is used to increase / decrease the location of the iterator based on the traversal direction. For example, ++ for `Iterator` moves the iterator from left to right. However, ++ for `Reverse_Iterator` moves the iterator from right to left (++ operator is defined in both classes differently).
- New node can be inserted to / deleted from either end (left or right) and in the middle where the location is given by the iterator.

You can access each element in the list using the iterator as follows:

```
for(Iterator it = Begin(); it!=End(); it++)
```

```
{
	std::cout << *it << ", ";
}
```

In `DoublyLinkedList.h`, all the required functions and data are defined, and the actual implementation of those functions should be in `DoublyLinkedList.txx`. You need to implement 20 functions marked as `//Todo`.

You **must not** modify **nor** add a new function in this header file. You will not submit the header file.

2. Simple text editor (50 pts)

You will implement a **line-based** simple text editor using the implemented doubly linked list. Using the simple text editor, you can insert or delete text line by line.

When a new input string (given as a pointer of char) is given, the entire string is stored as a line. You should use doubly linked list to manage text lines. Each text line has the line number, starting from 1.

For any given time, you can either print out the entire buffer (print out all lines), or print out only the current line. You can change your current line by move up and down.

`texteditor.cpp` is the implementation of `texteditor.h`, and it is what you need to modify and submit. You need to write code where `//ToDo` is written.

This is the list of 6 functions you need to implement:

- TextEditor() : constructor
- ~TextEditor() : destructor
- Insert(char*) : Insert a new line after the current position, and change the current position to the newly inserted line.
- Delete() : If the buffer is empty, print out the message "Buffer is empty". If not, you delete the current line and set the new current line as next line (if exists). If not, set the new current line as previous line.
- MoveUp() : Move current line up by one. If the current line is the first

line, do not move and print out the message “Beginning of the buffer”.

- MoveDown() : Move current line down by one. If the current line is the last line, do not move and print out the message “End of the buffer”.
- Run() : Run the text editor in interactive mode.

`main.cpp` is the main file to test your code. This file contains testing code to evaluate the correctness of your implementation. It consists of three parts: 1) Test doubly linked list, 2) Test text editor functions, 3) test interactive mode of text editor. If your implementation is correct, the result should be printed out as follows:

```
List 1 Size : 1
10,
--
List 1 Size : 2
20, 10,
--
List 1 Size : 3
30, 20, 10,
--
List 1 Size : 4
40, 30, 20, 10,
--
List 1 Size : 5
40, 30, 20, 10, 50,
--
List 1 Size : 6
40, 30, 20, 10, 50, 60,
--
List 2 Size : 6
40, 30, 20, 10, 50, 60,
--
Copy constructor
List 3 Size : 6
40, 30, 20, 10, 50, 60,
--
List 1 Size : 5
30, 20, 10, 50, 60,
--
List 1 Size : 4
30, 20, 10, 50,
--
List 2 Size : 6
40, 30, 20, 10, 50, 60,
--
List 3 Size : 6
40, 30, 20, 10, 50, 60,
--
```

```
First element of list 3 : 40
Second element of list 3 : 30
70 will be inserted after 30
40, 30, 70, 20, 10, 50, 60,
65 will be inserted before 70
40, 30, 65, 70, 20, 10, 50, 60,
10 will be deleted
40, 30, 65, 70, 20, 50, 60,
--
Print out the current buffer:
L1 : I am doing my assignment 2.
L2 : This assignment is not so easy.
L3 : But it is really fun.
L4 : I hope I can make it.
L5 :*KU CSE rocks!
Current(L5) : KU CSE rocks!
Move up
Current(L4) : I hope I can make it.
Move up
Current(L3) : But it is really fun.
Delete the current line
Current(L3) : I hope I can make it.
Print out the current buffer:
L1 : I am doing my assignment 2.
L2 : This assignment is not so easy.
L3 :*I hope I can make it.
L4 : KU CSE rocks!
Move down
Current(L4) : KU CSE rocks!
Insert new line
Print out the current buffer:
L1 : I am doing my assignment 2.
L2 : This assignment is not so easy.
L3 : I hope I can make it.
L4 : KU CSE rocks!
L5 :*Finally I made it~
>
```

Note that PrintAll() function prints out every line in the current text and the current line is marked with "*". The last test of main.cpp is interactive mode, so the cursor (>) will be waiting for your input. For example, if you simply press enter without typing in any text, then you will see the help message as follows:

```
>
No such command. Available commands are u(up), d(down), p(print),
pa(print all), r (delete), i(insert), and x(exit)
```

Here, you can enter a command and see the result interactively. The basic I/O with "u" and "x" implementations are already given in Run(), so you should

modify it so that you can handle all the other functions (d, p, pa, r, and i). For insert, the input prompt should be changed to "(Insert) > " and accept the user input. An example is shown below:

```
> pa
L1 : I am doing my assignment 2.
L2 : This assignment is not so easy.
L3 : I hope I can make it.
L4 : KU CSE rocks!
L5 :*Finally I made it~
> i
(Insert) > I am happy now
> pa
L1 : I am doing my assignment 2.
L2 : This assignment is not so easy.
L3 : I hope I can make it.
L4 : KU CSE rocks!
L5 : Finally I made it~
L6 :*I am happy now
```

3. Compile and submit

You must must zip `doublylinkedlist.txx` and `texteditor.cpp` and submit it online via blackboard. You can compile the code as follows:

```
> make
```

The output executable name is `assign_2`. You can run your code by simply type in this name in the terminal.

```
> assign_2
```

Good luck, and ask TAs and professor if you have any question.

Have fun!