CSCE 221 Assignment 2 Cover Page

First Name Last Name UIN

User Name E-mail address

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more on Aggie Honor System Office website: `http://aggiehonor.tamu.edu/`

Type of sources				
People				
Web pages (provide URL)				
Printed material				
Other Sources				

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.
On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.

Your Name Date

CSCE 221 Assignment 2

Due October 6th

Objective

This is an individual assignment which has three parts.

- Part 1 involves implementation of a **doubly linked list** and its templated version with the provided ADT and analyzing its complexity.
- Part 2 involves implementation of a class **Record,** and writing applications based on **doubly linked list**
- Report

Part 1 (30 points): Implementing Doubly Linked List

- **Program Instructions**

  Download the program `Starter Code` from eCampus to get an access to two separate folders.

1. Doubly linked list for integers
   (a) Contains a list node structure and associated functions. Doubly linked lists of integers can be constructed using the structure of a list node.
   (b) Most code is extracted from the lecture slides. An exception structure is added to make it more useable.
   (c) You need to complete the following functions in the `DLList.cpp`
      i. first
      ii. last
      iii. insert_first
      iv. insert_last
      v. remove_first
      vi. remove_last
      vii. insert_before
      viii. insert_after
      ix. remove_before
      x. remove_after
      xi. copy constructor
      xii. copy assignment operator
      xiii. move constructor and move assignment
      xiv. destructor
      xv. output operator (outside the class)

      The functions vii-x insert a node with an integer or remove a node before/after the current list node.

      Make sure the functions in xi. and xii. do a deep copy of the input list, that is, they have a real copy of each node (not a reference/pointer).

      For remove functions throw exceptions if remove is called on empty list or on header/trailer.
   (d) Type the following commands to compile the program.

      ```
      make clean
      make
      ```
   (e) The main program includes examples of creating doubly linked lists, and demonstrates how to use them. Type the following command to execute.

      ```
      ./run-dll
      ```

2. Templated DoublyLinkedList for general type

   (a) Convert the doubly linked list in the part 1 to a template, so it creates lists of other types, not only integer.

   (b) Follow the instructions below:

      i. **IMPORTANT: Templates should be declared and defined in a `.h` file.** Move the content of `DLList.cpp` and `DLList.h` to `TemplatedDLList.h`

      ii. Replace `int obj` by `T obj` in the class `DLListNode` so list nodes store general type `T` objects instead of integers. Later, when a `DLListNode` object is created, say, in the main function, `T` can be specified as an `int`, a `string` or a user-defined class.

      iii. To use a general type `T`, and use `DLListNode` and `DLList` of the general type `T`, you must change each type declaration.

         A. Replace variable declaration, input type and output type of functions `int` by the general type `T`, except for the `count` variable.

         B. Replace variable declaration, input type and output type of functions `DLListNode` by `DLListNode<T>`.

         C. Replace variable declaration, input type and output type of functions `DLList` by `DLList<T>`.

      iv. Assign the general default value `T()` to `T obj` of `DLListNode`, instead of the original 0 to `int obj`

      v. To use the general type `T` anywhere throughout the class `DLListNode` and `DLList`, you must declare (add) `template <typename T>` before classes and the member functions defined outside the class declaration where `T` is ever used

      vi. In each member function signature, replace `DLList::` by `DLList<T>::`

   (c) Type the following commands to compile the program.

```
make clean
make
```

   (d) The main program includes examples of creating doubly linked lists of "strings", and demonstrates how to use them. Type the following command to execute.

```
./run-tdll
```

3. Submit **DLList.cpp** and **TemplatedDLList.h** to mimir under assignment 2 part 1

4. There is no readme for part 1.

Part 2 (50 points): Application of Doubly Linked Lists

Part 2.0 (20 points): Implementation of a class Record

1. Declare a class `Record` for keeping information about a book.

   Declare class members for a book: **title**, **author's name**, **13-digit ISBN**, **publishing year**, and **edition number**.]

   Declare them as

   (a) title

   (b) author

   (c) ISBN

   (d) year

   (e) edition

2. Outside the class define

   (a) input `operator>>` to enter the record from the input file `Book.txt`.

   (b) output `operator<<` to print the record on screen.

   (c) equal-to `operator==` to compare two records by title, author's name and ISBN

   ```
   bool operator==(const Record& r1, const Record& r2) {
     /* complete the code here */
   }
   ```

   In a case when two records `r1` and `r2` have the same title, compare the author's name and ISBN. The function returns true when title, authors and ISBN match; otherwise, it returns false.

3. `Book.txt`: the input file contains unsorted book records in format given below (title, author's name, 13-digit ISBN (dashes are not required), publishing year, edition). You can add empty lines between records.

   ```
   Harry Potter And The Chamber Of Secrets
   J. K. Rowling
   9780439064873
   2000
   1st edition

   H is for Hawk
   Helen Macdonald
   9780802123411
   2015
   1st edition
   ...
   ```

4. Test your class in the main function using input data from the file `Book.txt`.

Part 2.1 (30 points): Library Management System

- You should implement a library management system to store books. The system stores each book title, author's name, 13-digit ISBN, publishing year, and edition number. It is possible to have the same title and author's name for a book if there are more than one edition.
- Your library management system should provide a friendly interface for users to create a book database and search in this database.
  - The user will be asked to input the title to start searching.
  - If the program does not find a book with the requested title, the user will be asked to add this title to the database, and he/she needs to provide all the required book information.
  - If more than one book have the same title and author's name, these records will be displayed, and the user needs to decide which book edition to select.
  - Finally, the program will display the book.
- **The Data Structure**
  - To speed up the search in the library management system, the data will be stored in a vector of 26 sorted doubly linked lists. The sorting is done in alphabetical order with respect to the first letter of the book title, a letter is from A to Z.
    - For example, the eighth element of the vector, i.e. the eighth doubly linked list, `v[7]`, corresponds to the letter 'H'. I may contain, for instance, the following book records
      - "*H is for Hawk*", Helen Macdonald, 978-0802123411, 2015, 1st edition
      - *"Harry Potter And The Chamber Of Secrets*", J. K. Rowling, 978-0439060000, 2000, 1st edition
      - *"Harry Potter And The Chamber Of Secrets*", J. K. Rowling, 978-0439060001, 2000, 2nd edition
  - Again, to speed up the search, each doubly linked list must be maintained in sorted order by title, author name, and ISBN (in this order). You can treat ISBN as a string.

Submission

- Submit **TemplatedDLList.h, Record.cpp,** and **Library.cpp** to mimir under assignment 2 part 2
- There is no readme for part 2

Report (20 points)

Follow the report instructions found on eCampus under the "Assignment Cover Page and Report Format" section, with the requirements below:

- In the algorithm description section,
  - briefly describe class Record implementation and operator overloading and their **time complexity analysis***
  - briefly describe the functions you implemented in Part 1 and their **time complexity analysis***
  - briefly describe implementation of functions in the Part 2 and their **time complexity analysis***
  - the average **complexity analysis*** runtime for insert and search functions. Assume that the average length of each linked list is the same.
- **Complexity Analysis* –** provide a running time function for `n` records and express it in terms of big-O notation.
- Include in the report the screenshots as the evidence of testing the functions implemented in this assignment for correctness.
- Submit Report to ecampus