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

Spring 2020

Objective

This is an individual assignment which has two parts.

1. Part 1: C++ implementation of the doubly linked list class `DLList` for integers.
2. Part 2: C++ implementation of the templated doubly linked list class `DLList` for generic types based on the provided supplementary code.
3. Part 3: C++ implementation of `MinQueue` data structure that can store *comparable elements.*

README File

(5 points) For each part (1,2 3) create a text file, called README, in which you provide:

- Your First Name, Last Name, UIN, Section Number, User Name, E-mail address
- State the Aggie Honor statement:

  I certify that I have listed all the sources that I used to develop the solutions and code 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

- List any resources used such as webpages (provide URL). Do not mention the textbook and discussions with the Instructor, TA, or Peer Teachers.
- List any known problems with the assignment you are turning in. For example, if you know your code does not run correctly, state that. This should be a short explanation.
- Provide a short description or pseudocode.
- Write how you tested your program(s) for correctness and how you used exceptions for the error handling. Provide all tests done to verify correctness of your program in this README file. Give a short explanation why you selected such tests.
- Submit to eCampus an electronic version of the README file along with your C++ code for the problems listed below and (optional) a hard copy to your TA.
- The assignment will be graded focusing on: program design, correctness.
- Your programs will be tested on a CSE Linux machine using a TA's input file.

Part 1: The class DLList (presentation in labs by February 24/25)

1. Max points: 65 (Part 1) + 5 (README) = 70 points.
2. Untar supplementary code `221-A2-20a-code.tar`. Use the 7-zip software to extract the files on Windows, or use the following command on Linux.

   ```
   tar xfv 221-A2-20a-code.tar
   ```

   You should see two new directories: **DLList-class (for Part 1) and Templated-DLList-class (for Part 2).**
3. `DLList` for integers
   (a) Most of the code is extracted from the lecture slides. An exception structure is defined to complete the program.
   (b) You need to complete the functions which are declared in the header file `DLList.h`.
   (c) Type the following commands to compile the program

   ```
   make
   ```

   (d) The main program includes examples of creating doubly linked lists, and demonstrates how to use them. Type the following command to run the executable file:

   ```
   ./run-dll
   ```

   (e) Test the doubly linked list functions in `main`.

Part 2: The templated class DLList (due February 28 to eCampus together with Part 1)

1. Max points: 25 (Part 2) + 5 (README) = 30 points.
2. Implement a templated version of the class `DLList` and test the functions for correctness. Follow the instructions below:
   (a) Templates should be declared and defined in a header `.h` file. Move the content of `DLList.cpp` and `DLList.h` to `TemplatedDLList.h`
   (b) Replace `int obj` by `T obj` in the `struct DLListNode` so the list nodes store generic objects of type `T` instead of integers. Later on, when a `DLListNode` object is created, say, in the main function, `T` can be specified as a `char`, `string` or a user-defined class.
   (c) To create a templated class with a generic type `T`, you must replace a declaration/return type `int` by `T` (except for the `count` variable).
      i. To use the generic type `T`, you must change each type declaration.
      ii. Use the generic type `T` anywhere throughout the class `TemplatedDLList`.
   (d) Add the keyword `template<typename T>` before a class declaration.
   (e) In each member function signature, replace `DLList::` by `DLList<T>::`
   (f) If a member function is defined outside the class declaration, change the function signature, that is, replace `DLList::` by `TemplatedDLList<T>::`
   (g) To use the generic type `T` anywhere throughout the class `DLListNode` and `DLList`, you must declare (add) `template<typename T>` before classes and member functions defined outside the class declaration.
3. Compile and run the generic version in a similar way as for `int` type. Type the following commands to compile the program.

```
make
```

4. The main program includes examples of creating doubly linked lists of `string`, and demonstrates how to use them. Type the following command to run the executable.

```
./run-tdll
```

Part 3: MinQueue data structure based on templated DLList (due March 7 to eCampus)

- Max points: 45 (Part 3) + 5 (README) = 50 points
- The `MinQueue` data structure should store *the comparable elements* that support the queue operations: `enqueue(x)`, `dequeue()`, `size()`, `is_empty()`, and, in addition, the `min()` operation that returns (but not deletes) the smallest value currently stored in the queue.
- Use the adapter design pattern for implementation of `MinQueue` that work together with the templated class `DLList` defined in Part 2. The runtime worst case of all operations except `min()` should be *constant,* $O(1)$.
- The implementation details of the `MinQueue` operations, justification of their running time, and tests for correctness should be provided in the README file.

What to submit to eCampus?

- Create a separate directory for the Part 1 and Part 2 that includes: `DLList` source code for `int` and generic types (templated version), and the README files. Make a zip file from these two directories. **Due February 28 to eCampus.**
- Create a separate directory for the Part 3 that includes: `MinQueue` source code and the README file. Make a zip file from this directory. **Due March 7 to eCampus.**