COSC2436 Programming Fundamentals III /ITSE2445 Data Structures

The submission opens two days before the due day, and closes right at the due time on due day. One submission is allowed per project. It is students' responsibility to submit timely and correctly. Students MUST start each project when it is first discussed in class in order to complete it on time. All projects are individual projects. Project due days may be changed to better align with lectures.

If one of the following is true, not credits will be given:

- The project is late.
- The algorithm or class design is missing.
- The project has errors.
- There are no comments (Javadoc format required) at all in your code.
- Wrong files are submitted.
- **A project is a copy and modification of another student's project. (Both will receive 0.)**

Files to be submitted through Canvas:

- The UML design (UML class diagram)
- All source codes and driver programs
- All supporting files if any

Software Development Project Rubric: (Design is worth 20%; the rest is worth 80 %.)
Analysis: There will be no credit if the software doesn't meet customer specification at all.

- Does the software meet the exact customer specification?
- Does the software read the exact input data and display the exact output data as they are shown in sample runs?
- Does each class include all corresponding data and functionalities?

Design: There will be no credit if the design is missing.

- Is the design (a UML class diagram) an efficient solution?
- Is the design created correctly?

Code: There will be no credit if there are errors.

- Are there errors in the software?
- Are code conventions and name conventions followed?
- Does the software use the minimum computer resource (computer memory and processing time)?
- Is the software reusable?

Debug:

- Are there bugs in the software?

Documentation: There will be no credit if comments are not included.

- Are there enough comments included in the software?
- Class comments must be included before a class header.
- Method comments must be included before a method header.
- More comments must be included inside each method.
- All comments must be written in Javadoc format.

Project 2 ADT Character String **Due: See calendar**

Implement the ADT character string as the class *LinkedString* by using a doubly linked list of characters.

ADT Character String Specification:

Specify operations to

- allocate a new character linked list so that is represents the sequence of 0 characters currently contained in the character array argument. *(LinkedString()),*
- allocate a new character linked list so that is represents the sequence of characters currently contained in the character array argument. *(LinkedString(char[])),*
- initialize a new character linked list so that it represents the same sequence of characters as the argument *(LinkedString(String)),*
- return the char value at the specified index. The first character in this linked character string is in position zero. *(char charAt(int)),*
- concatenate the specified linked character string to the end of the linked character string (*LinkedString concat(LinkedString)),*
- return true if, and only if, length() is 0 *(boolean isEmpty()),*
- return the length of this linked character string *(int length()),*
- return a new linked character string that is a substring of this linked character string (*LinkedString substring(int, int)).*

Exceptions should to be considered when operations are designed.

ADT Character String Design:

Complete a UML diagram to include all classes that are needed to meet the specifications. An interface class is usually defined to include all operations. A class implementing this interface provides implementation details. Exception classes are define if needed. Implement *LinkedString* so that it is consistent with the *String* class. For example, character position start at 0. Also, keep track of the number of characters in the string; the length should be determined without traversing the linked list and counting.

ADT Character String Reference-based Implementation:

Implement all classes included the design. Javadoc comments should be included during this activity. A reference – based implementation is required in this project. A doubly linked list must be used. All characters should be linked each other. Use Generics, develop classes and interfaces and defer certain data-type information until you are actually ready to use the class or interface.

ADT Character String Test/Debug:

Complete a driver program to test all operations in the design. Method decomposition must be used to break method *main* into parts. All testing data must be included in a text file. Here is a template of method *main*:

```
public static void main(String[] args) {
       //Create instances of the ADT Linked character string using all three constructors
       …
}
```

ADT Character String Documentation:

Check the entire project, and complete /revise Javadoc comments if needed.