ASSIGNMENT INSTRUCTIONS

1. Assignment 04: **75 points w/ 0 E.C. points**
2. Due Date & Time: **11-04-2022 at 11:55 PM**

WHAT TO SUBMIT

1. Assignment Report
2. Code

HOW TO SUBMIT AND THE RULES TO FOLLOW

- Submit via Canvas, the Assignment Submission section
- Please refer to Assignment 01 for the Assignment Guidelines
- Please follow the Assignment Report Template
- Please follow the Course Policy on Student Conduct and Academic Honesty

PERFORMANCE TRACKER		
ASMT	GRADE	YOUR GRADE
CANVAS	05	
01	20	
02-PREPARATION	25	
02	75	
03	75	
MIDTERM EXAM 01	25	
04	75	
TOTAL	300	

A: 90-100% **B**: 80-89% **C**: 70-79% **D**: 60-69% **F**: 0-60%
The course grader provides feedback to your assignments on Canvas.

ABOUT

This assignment contains several small problems. We will need the starter code and the output (part E) provided here:

- **Download**: http://csc220.ducta.net/Assignments/Assignment-04-Code.zip
- We must use the provided starter code and follow the assignment instructions to get credit.

The programming techniques and the data structures which we practice in this assignment is very important. We need to master them in order to understand the advanced part of CSC 220 and to prepare ourselves for success in the coming advanced courses starting with CSC 340.

PART A – Introduction to Sorting, **9 points**

Please use this array of integers for the A.1, A.2, and A.3 problems: `5 2 3 4 7 9 6 2 8 3 1 5`

- Use pen & paper to show our work and answers. We can scan/snapshot our work and include the images in our report.
- Use code to demonstrate our approaches and solutions. Submit code and include screenshots of output in our report.
- Each of these problems (A.1, A.2, and A.3) is worth **3 points**.

1. Show the contents of the array each time a **selection sort** changes it while sorting the array into **ascending order**.
2. Show the contents of the array each time an **insertion sort** changes it while sorting the array into **ascending order**.
3. Show the contents of the array each time a **Shell sort** changes it while sorting the array into **ascending order**.

PART B –Sorting, **11 points**

1. --- **3 points** --- Suppose we want to find the largest entry in an unsorted array of n entries. Algorithm A searches the entire array sequentially and records the largest entry seen so far. Algorithm B sorts the array into descending order and then reports the first entry as the largest. Compare the time efficiency of the two approaches. *Coding is not required but highly recommended.*

2. --- **8 points** --- Consider an `n by n` array of integer values. Write an algorithm to sort the rows of the array by their first value.

   The starter code for this problem is provided in the **Assignment-04-Code.zip** archive. Our output must be **identical** to the output to the right.

NO User Input

```
The array is initially
    1  2  3  4  5
    3  4  5  1  2
    5  2  3  4  1
    2  3  1  4  5
    4  2  3  1  5

The array after sorting is
    1  2  3  4  5
    2  3  1  4  5
    3  4  5  1  2
    4  2  3  1  5
    5  2  3  4  1
```

PART C – Queues, Deques, and Priority Queues, **15 points**

1. --- **5 points** --- After **each of the following statements** executes, what are the contents of the **queue**? Please explain.

```
QueueInterface<String> myQueue = new LinkedQueue<>();
myQueue.enqueue("Jane");
myQueue.enqueue("Jess");
myQueue.enqueue("Jon");
myQueue.enqueue(myQueue.dequeue());
myQueue.enqueue(myQueue.getFront());
myQueue.enqueue("Jim");
String name = myQueue.dequeue();
myQueue.enqueue(myQueue.getFront());
```

2. - **5 pts** - After **each of the following statements** executes, what are the contents of the deque? Please explain.

```
DequeInterface<String> myDeque = new
LinkedDeque<>();
myDeque.addToFront("Jim");
myDeque.addToFront("Jess");
myDeque.addToBack("Jen");
myDeque.addToBack("Josh");
String name = myDeque.removeFront();
myDeque.addToBack(name);
myDeque.addToBack(myDeque.getFront());
myDeque.addToFront(myDeque.removeBack());
myDeque.addToFront(myDeque.getBack());
```

3. - **5 pts** - After **each of the following statements** executes, what are the contents of the priority queue? Please explain.

```
PriorityQueueInterface<String> myPriorityQueue =
new LinkedPriorityQueue<>();
myPriorityQueue.add("Jim");
myPriorityQueue.add("Josh");
myPriorityQueue.add("Jon");
myPriorityQueue.add("Jane");
String name = myPriorityQueue.remove();
myPriorityQueue.add(name);
myPriorityQueue.add(myPriorityQueue.peek());
myPriorityQueue.add("Jose");
myPriorityQueue.remove();
```

It is OK to assume that the alphabetically earliest string has the highest priority.

PART D – Queue and Deque, Circular Doubly Linked Chain, **20 points**

Use a circular doubly linked chain to **implement** the ADT deque.

In a **doubly linked chain**, the first and last nodes each contain one null reference, since the first node has no previous node and the last node has no node after it. In a circular doubly linked chain, the first node references the last node, and the last node references the first. Only one external reference is necessary—a reference to the first node—since we can quickly get to the last node from the first node.

NO User Input

```
Empty deque: true
[FRONT] Jerry << Tom << Minnie << Mickey << ...
 [BACK] Sylvester >> Goofy >> Donald >> ...

Empty deque: true
Sayōnara
-> removeFront found deque empty
-> removeBack found deque empty
```

The code for this problem is provided in the **Assignment-04-Code.zip** archive. Our output must be **identical** to the output to the right.

PART E – Priority Queue, **20 points**

The San Francisco State University's One Stop Student Services Center asks us to recommend solutions for their service lines.

- The starter code for this problem is provided in the **Assignment-04-Code.zip** archive.
- Our output must be **identical** to the **complete** output provided in the ZIP archive: ***PartE-The_Complete_Sample_Run.pdf***
- *The right table is a portion of the output for preview purposes. It is NOT the complete output.*
- *It is a good idea to analyze the complete output thoroughly before programming a solution.*

NO User Input

```
------------------------------------------------------------
            SFSU ONE STOP STUDENT SERVICES CENTER
------------------------------------------------------------
Priority: realistic (provided by supervisors)
          Minnie    Mouse     1001      3.90       10       15
          Mickey    Mouse     1002      3.70        1       17
          Goofy     Dog       1007      2.30       17        1
          Milo      Dog       1004      3.70        7       17
          Pluto     Dog       1005      3.70        7       17
          Daisy     Duck      1003      1.70        1       17
          Donald    Duck      1006      3.10        5        2
```