SAN DIEGO STATE UNIVERSITY

Department of Computer Science
CS 210 Data Structures

Program 1 Phase 3 Instructions

Student Objective

In this assignment, you will develop two implementations of a Priority Queue: one ordered and one unordered. Both implementations must adhere to specific criteria, ensuring removal operations consistently return the object of the highest priority that has been in the queue the longest. Additionally, the FIFO order must be maintained within each priority level.

This assignment will progress through three distinct phases. Phase 1 involves the implementation of two Array classes: `UnorderedArrayList` and `OrderedArrayList`. Following Phase 1, Phase 2 focuses on the development of a Priority Queue specific to each array class type. Finally, Phase 3 will entail conducting an analysis of your written code, evaluating aspects such as time complexity, space complexity, and overall clarity.

As you tackle this first program, remember it's your chance to grasp the basics of testing, finding edge cases, and mastering debugging. Embrace the opportunity to learn test-driven development—it's the perfect starting point for your journey in this course!

Phase 3 Instructions

Congratulations on completing the challenging part of the program! Now, you'll be analyzing your code, an essential aspect for continual improvement often demanded in industry. This analysis will take place within the Gradescope assignment for Phase 3.

You're tasked with uploading each of your implementations (the `.cpp` files) and providing detailed explanations for various methods' time and space complexities. For each method, such as add(int data) or clear(), you'll need to explain their time and space complexities using Big O notation. Additionally, justify why the complexities are as they are and assess whether they're optimal, referencing your code as evidence.

Once you've covered the ordered array list complexities, you'll move on to the unordered array list complexities, followed by ordered and unordered priority queue complexities. Again, upload the corresponding files and provide thorough explanations for each method's time and space complexities.

Finally, you'll delve into real-world analysis, exploring applications where priority queues are utilized. Discuss the benefits and drawbacks of using priority queues in various scenarios, showcasing your understanding of their practical significance.

Cheating Policy

There is a zero tolerance policy on cheating in this course. You are expected to complete all programming assignments on your own. Collaboration with other students in the course is not permitted. You may discuss ideas or solutions in general terms with other students, but you must not exchange code. During the grading process I will examine your code carefully. Anyone caught cheating on a programming assignment (or on an exam) will receive an ”F” in the course, and a referral to Judicial Procedures.