COMP 2540 Data Structures and Algorithms

University of Windsor

School of Computer Science

Assignment 3: Due (July 6 to 10) 2020

This assignment is to understand the Queue ADT and Priority Queue ADT. We discussed both Queue and Priority Queue in the lectures. The priority queue is similar to a regular queue; the difference is each element in the priority queue is associated with a "priority" value. It means that an element with higher priority is considered before an element with lower priority. If two elements have equal priority value, they are considered as per insertion order.

Task 1:

Implement a regular Queue ADT using the following two different representations:

1. Array
2. Linked List

Your Queue Class should allow any programmer to store any element (data) without understanding or knowing the details of the underlying representation. Your Queue ADT should provide the following operations:

- enqueue(data): inserts an element at the end of the queue.
- dequeue(): removes and returns the element at the front of the queue.
- peek(): returns the element from queue without removing it.
- getSize(): returns the number of elements stored.

- isEmpty(): return false if no elements are stored.
- isFull(): return true if queue is full.
- display():prints all the elements in the queue.

Task 2:

Modify the above code to implement the Priority Queue ADT using the following two different representations:

1. Array
2. Linked List

Your Priority Queue Class should allow any programmer to store any element (data) without understanding or knowing the details of the underlying representation.

Your Priority Queue ADT should provide the following operations.

- enqueue(element, priority): insert an element to the queue appropriate position in the queue based on the associated priority. If the priority parameter is missing, assign to the element 10 as default priority, which is the lowest priority.
- dequeue(): removes and returns the entry with highest priority from the queue, or null if the priority queue is empty. If two elements have the same priority, they are considered according to their order in the queue.
- peek(): return the element to be considered from the queue without removing it.
- size(): returns the total number of elements in the queue.
- isEmpty(): returns true if the queue is empty.
- isFull(): returns true if the queue is full.
- display(): prints all the elements in the queue with their priority.

NOTE:

1. You should test the following test cases in your program.
    a. Task 1:

        Test 1: enqueue → Java, Python, XML, HTML, CSS, LISP
        display, dequeue, display, enqueue (JSON), display.

        Test 2: do your own test case for different set of data input.

    b. Task 2:

        Test 1: enqueue→("Java", 2), ("Python", 1), ("XML", 4), ("HTML", 2), ("CSS", 3)
        display, dequeue, display, enqueue (JSON, 3), display.

        Test 2: do your own test case for different set of data input.

        Test 3: test only with default priority value (i.e 10). Explain the relationship with task 1.

    Test each case and create a make file. (submit your make file with your source code.)

2. You **CAN NOT** use inbuild functions for Queue ADT or Priority Queue ADT. If you do so, you will get **ZERO**. Other required inbuild functions are allowed to use.

Requirements:

I. **UNDOCUMENTED OR IMPROPERLY DOCUMENTED** code will automatically lose 50% marks.

II. Readability: The code is exceptionally well organized and very easy to follow.

III. **PLAGIARIZED** work will not be graded and receive a mark of **ZERO** and reported according to the Senate bylaws.

Submission:

I. Your assignment must be RECEIVED by the due date and time. Late assignment submissions are NOT accepted will receive a mark of ZERO.

II. Before the submission, you must contact your GA/TA through your lab MS team channel for the feedback.

III. Submit your work through Blackboard: Source code

Rubric:

5 - The program works and meets all the above requirement.

4 - The program works, fair readability, most of the code could be reused and fair documentation.

3 - The program works, fair readability, most of the code could be reused and NO documentation.

2- The program produces some results but does not display them correctly, poor readability and NO documentation.

1- The program produces compilation error, poor readability, and NO documentation

0 - Did not complete any task.

PLUS

*** 1 - Contact the GA/TA to answer the questions. (attendance + students' involvement)