BPDC, Dubai – II Semester, 2020-2021

Course No: CS F211 **LAB EXERCISE 4** **Course Title: DSA**

Date: Feb 22 2021

Implement subroutines that deploys the basic operations (prototype given below) of a circular queue. Further, copy the same for linear queue from last week's assignment. You are provided with a program circularQueue.c that tabulates the number of successful enqueue/dequeue operations in files circular_Qcount.txt and linear_Qcount.txt respectively, for each of those test cases provided with. You may assume the queue only to deal with positive integers. Plot testCaseSize Vs Qcount both for linear and circular implementations (you would have two plots in a single XY-frame).

The dynamically allotted array *circular_queue*[] (*linear_queue*[]) refers to the memory allotted for the abstract data type circular queue (linear queue) along with the corresponding index variables *frontC (frontL)* and *rearC (rearL)*, already being defined and initialized to 0. The size of the array is in variable *circular_queue_size (linear_queue_size)*. You may use the same in the implementations of the functions below.

int isLinearQueueEmpty() //returns 1 if the linear queue is empty and 0 otherwise

int isLinearQueueFull() //returns 1 if the linear queue is full and 0 otherwise

linearEnqueue(x) //enqueues the value x to the rear end of the linear queue, returns -1 if in case the queue is full

linearDequeue() //dequeues the value x from the front end of the linear queue, returns -1 if in case the queue is empty

int isCircularQueueEmpty() //returns 1 if the circular queue is empty and 0 otherwise

int isCircularQueueFull() //returns 1 if the circular queue is full and 0 otherwise

circularEnqueue(x) //enqueues the value x to the rear end of the circular queue, returns -1 if in case the queue is full

circularDequeue() //dequeues the value x from the front end of the circular queue, returns -1 if in case the queue is empty

(a) Consider a huge array (say of size N) A[] containing a few positive integers and -1 elsewhere. Let the number of positive integers in A[] be n and let them be stored in a specific format as detailed below.

- A[0] is -1 and A[1] is a positive number
- If A[i] is a positive number and i>1 then A[⌊i/2⌋] is a positive number where ⌊ ⌋ is the floor function

Write a program (in C,C++, or Java) that would list all those positive numbers in the array A[] in O(n) time, using a circular queue. You could focus on building a program that would read input from the user to begin with and afterwards, extend the same to meet the following requirement.

You are provided with a file binaryTreeArray.txt containing 10 test cases, where each test case is a line starting with the size N of the array, followed by elements (N of them) in the array. You are supposed to read the value of N to begin with, **allocate sufficient amount of memory (dynamically) for your array variable (this has to be declared as an integer pointer,** refer previous programs if required) and then read the input and process.