# Real Time Systems, January 2024



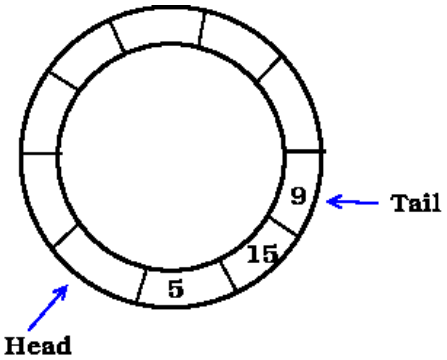
## Add README.txt against you submission on Implementation of a circular queue to be used by multiple threads of a process

**Opened:** Wednesday, 28 February 2024, 12:00 AM
**Due:** Sunday, 10 March 2024, 11:59 PM

✔ Done

---

Let there be a circular queue (holding **at most 10** integer items) **to be used by** multiple threads of a process, say **m** number of producer threads (***running void *producer(void *data)*** function) and **n** number of consumer threads (***running void *consumer(void *data)*** function). Each of the **m** producer threads, in every iteration of an infinite loop, adds one or more items (**random** integers) to the queue. A consumer thread, on the other hand, in an infinite loop, "consumes" one or more items (integers) from the queue.




Write the program ***producerConsumer.c*** to implement the above scheme.

- There should be a manager thread which, depending on user's choices, will **add** and/or **delete producer** and/or **consumer** threads. That is, the manager thread should be **interactive** with the user to collect his/her choice(s). You have to decide what all choices this manager thread should offer to the user.
- At a time a **producer** thread may add one or more (maximum 10) items to the queue and a **consumer** thread too may "consume" one or more (maximum 10) items from the queue.
- producer and consumer threads should produce appropriate output so that the user can validate their proper functioning.
- You may like to adopt some mechanism so that the output of the threads is not too fast to be read by the user.
- The program should include ***enQ(), deQ()*** functions having typical functionalities. **You have to decide the proper signature for these functions. You may require to write additional helper functions (like noOfFreeSpacesQ() -** to get the number of free spaces in the queue, etc). You decide the signature of such helper functions too.
- Since the circular queue is shared by both producer and consumer threads, there are possibilities for race conditions to occur. Ensure that there is **no race condition.**
- A **Producer** thread cannot add item(s) if there is not enough space in the queue and should wait for consumer thread(s) to "consume" item(s).
- A **Consumer** thread, on the other hand, cannot "consume" item(s) if there is not required number of items in the queue, and should wait for producer thread(s) to add item(s).
- producer and consumer threads **should avoid busy waiting** as far as practicable.

**Your programs must be user-friendly and well-documented!**

## Submission status

| | |
|---|---|
| **Attempt number** | This is attempt 1. |
| **Submission status** | Submitted for grading |
| **Grading status** | Not graded |
| **Time remaining** | Assignment was submitted 2 days 11 hours early |
| **Last modified** | Friday, 8 March 2024, 12:09 PM |
| **File submissions** | pcdraft.c 8 March 2024, 12:26 AM<br>Readme.txt 8 March 2024, 12:26 AM |
| **Submission comments** | Comments (0) |

◄ Producer - Consumer Threads (version 2)

Jump to...

Fresh submission on Implementation of a circular queue to be used by multiple threads of a process ►