CSE213L: DSA Lab

LNMIIT, Jaipur

Training Set 02

In this training, students will learn basic ADT operations on queue and deque. In addition, class will learn about a way to collect functions into modules. A convenient arrangement for this is based on files and directories. The functions that are closely related to one-another may be grouped together in a file (or module). The idea is to have strong cohesion and commonality of purpose among the functions in a single module/file.

In our training program in this set, it is organised around a deque and a queue. To help you start, we give a partial implantation of a deque. There are three files provided to you. You may use them to quickly start working on the tasks described below.

Training Set 02: Task 01

Read K&R book for more information about the header files and their purposes. Other good C books also will have this information. There are many features which we have not used in this code, but over time as a professional in the discipline, you need to learn them. However, the given code is a good start for the present.

1. Write the code for the functions in module `deque.c` that are incomplete.
2. Next, construct a header file `queue.h` with ADT functions needed by program `main.c`. At present, function `main()` is using an implementation of deque for needs that are better satisfied through ADT interface for a queue.
3. Create a file `queue.c` in which you will implement functions in `queue.h`. The new code needed is very simple. The functions will simply forward the call to the appropriate functions of the deque.
4. In your demonstration of this task to your tutor, your program must demonstrate:
    a. function `main()` that includes interface to queue, but not to deque. It should perform the same tasks that the present code does.
    b. You queue functions `joinQ()` and `leaveQ()` should be implemented by calls to the deque functions that are unimplemented in the given code.

Training Set 02: Task 02

Study and learn the well-known algorithm for finding prime numbers called Sieve of Eratosthenes (https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes) before you lab session. You should be able to write code to compute primes in range 1 to 99999 using the implementation of queue in Task 01. For this you may need to set SIZE to value 100001.

1. Initially, fill the queue with all values from 2 to 100000. This is your initial run (sequence) of candidates for primes. Insert a special marker value into the queue to separate it from the next run of (surviving) candidates that you will insert in the queue next. We will now set up for finding the prime values. In each iteration, a prime will be selected and used to eliminate unsuccessful candidates.

2. Each run of the sieve will read the candidate values from the queue till the run of candidates ends. This end will be noted when the special separator value is read.
    a. First number read from a run is a prime and it is printed on the output stream (`stdout`).
    b. After that the numbers are read and inserted into the queue to build the next (new) run of candidates for primes. The multiples of the prime number are removed and eliminated. The numbers not eliminated are added to the queue.
3. The process ends when we can not build a new run of candidates for primes. No new candidate is available.

Training Set 02: Task 03

Modify the implementation of deque as follows.

We will change the purpose and meaning of indices `left` and `right`. These will be indices where the new data can be placed. Note this is different from how we used it in the given implementation (and in our textbook), In the implementation given to you, these index point at the position where data is already stored.

Indices `left` and `right` will point at the positions in the array that are available for the next data. For example, function `init()` can set `left = 8` and `right = 9`. In this new arrangement, clearly the array can only hold SIZE – 1 values and no more.

Re-implement all functions for this new approach. After a correct implementation of the module, Task 02 should work correctly *without any* modification in files `deque.h`, `queue.c`, `queue.h` and `main.c`.

That is all for today.

Remember, we impose a strict limit on the number of tasks marked in a session. *This limit is 3 tasks*. However, tasks for Training Set 01 may be shown in the second session as some lab sessions were not started on their first timetable schedule.