































Practice Arena

Practice problems aimed to improve your coding skills.

-  PRACTICE-02_SCAN-PRINT
-  PRACTICE-03_TYPES
-  LAB-PRAC-02_SCAN-PRINT
-  LAB-PRAC-01
-  PRACTICE-04_COND
-  BONUS-PRAC-02
-  LAB-PRAC-03_TYPES
-  PRACTICE-05_COND-LOOPS
-  LAB-PRAC-04_COND
-  LAB-PRAC-05_CONDLLOOPS
-  PRACTICE-07_LOOPS-ARR
-  LAB-PRAC-06_LOOPS
-  LAB-PRAC-07_LOOPS-ARR
-  LABEXAM-PRAC-01_MIDSEM
-  PRACTICE-09_PTR-MAT
-  LAB-PRAC-08_ARR-STR
-  PRACTICE-10_MAT-FUN
-  LAB-PRAC-09_PTR-MAT
-  LAB-PRAC-10_MAT-FUN
-  PRACTICE-11_FUN-PTR
 -  Circular Queue
 -  Primes are here again
 -  The Clones of the Clones
-  LAB-PRAC-11_FUN-PTR
-  LAB-PRAC-12_FUN-STRUC
-  LABEXAM-PRAC-02_ENDSEM
-  LAB-PRAC-13_STRUC-NUM
-  LAB-PRAC-14_SORT-MISC

Circular Queue

PRACTICE-11_FUN-PTR

Queues are very widely used in Computer Science (see week 10 lab problems Tuesday problem 1 and Thursday problem 1 - please understand those questions before solving this question). However, when implementing queues using arrays, managing the limited memory available becomes an issue. Suppose we have an array of size 5 and wish to implement a queue using this array.

Suppose the operations given to us are (E = enqueue, D = dequeue, X = terminate)

E 1

E 2

E 3

E 4

E 5

D

D

Initially the array is empty (* denotes an empty location)

[* * * * *]

Then after the five enqueue operations, the array looks like

[1 2 3 4 5]

whereas after the next two dequeue operations, the array looks like

[* * 3 4 5]

the first two positions are no longer a part of the queue. Now if there are two more enqueue operations

E 6

E 7

then we have a problem in our hands. There is no more space to the right of the array (new elements in the queue always go to the back of the queue) so we can do either of the following three things

1. Refuse to enqueue the two new elements: although simple, this is a waste of space since there are two empty locations in the array
2. Shift the array to fill up the empty spaces: shift all entries two locations left so that the array looks like
[3 4 5 * *]
Now there is space to enqueue 6 and 7. However, this shifting business can take up a lot of time (imagine a queue with 3 million entries)
3. Reuse the space to the left using a circular queue: this is what we will implement in this question.

A circular queue is very efficient in reusing space in that it cycles around if the array has no more space to the right. Thus, in a circular queue, E 6 would result in

[6 * 3 4 5]

and then E 7 would result in

[6 7 3 4 5]

If we now see a D instruction, we would pop 3 and the array pop an element from the front of the queue which is the element 3, and the array would look like

[6 7 * 4 5]

If we see 3 more D instructions, elements would keep getting popped from the front of the queue and the array would look like

[6 7 * * 5]

[6 7 * * *]

[* 7 * * *]

If we now see the instruction E 8, the array will get added to the back of the queue

[* 7 8 * *]

If we pop two elements now, the array would become all empty again

[* * * * *]

If we add a new element now E 9, it would get added to the back of the queue which gets reset to the first element of the array

[9 * * * *]

Remember, dequeuing from an empty queue results in an "UNDERFLOW" error and enqueueing to an already full queue results in an "OVERFLOW" error.

You have to implement enqueue and dequeue operations in a circular queue as described above and print the state of the array after every operation (or else print the error message). The first line of the input will tell you the size of the array you should use.

 **Start Solving!** (/editor/practice/6212)