CS 300 Data Structures
Problem Set # 14 - Queue

1. A queue follows ___________
   a) FIFO (First In First Out) principle
   b) LIFO (Last In First Out) principle
   c) Ordered array
   d) Linear tree

2. If the elements "A", "B", "C" and "D" are placed in a queue and are deleted one at a time, in what order will they be removed?
   a) ABCD
   b) DCBA
   c) DCAB
   d) ABDC

3. Queues serve major role in ______________
   a) Simulation of recursion
   b) Simulation of arbitrary linked list
   c) Simulation of limited resource allocation
   d) Simulation of heap sort

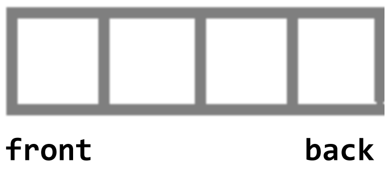
4. In a circular queue, how do you increment the rear end of the queue?
   a) back++
   b) (back+1) % CAPACITY
   c) (back % CAPACITY)+1
   d) back--

5. What is the time complexity of enqueue operation?
   a) O(logn)
   b) O(nlogn)
   c) O(n)
   d) O(1)

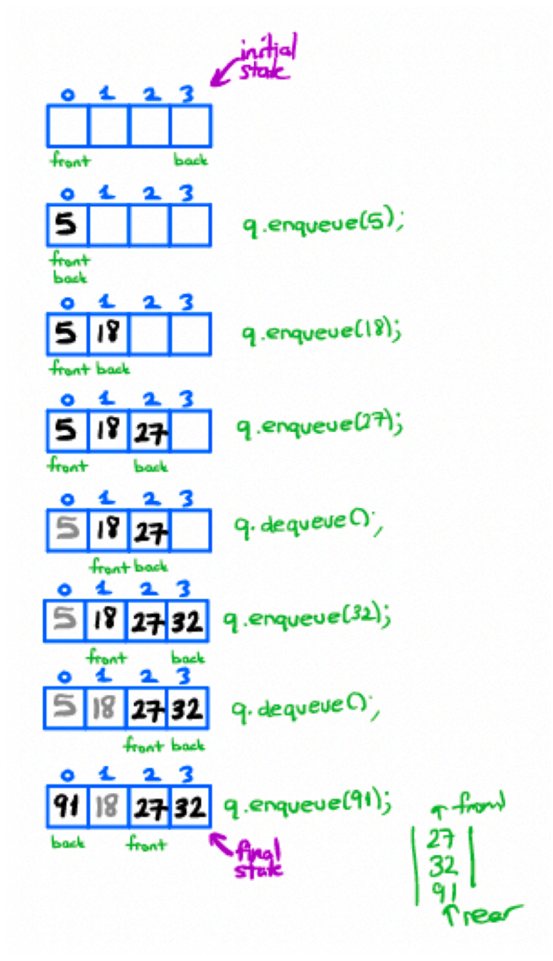
6. What is the need for a circular queue?
   a) effective usage of memory
   b) easier computations
   c) to delete elements based on priority
   d) implement LIFO principle in queues

7. Given an empty queue Q implemented with circular array, what does it look like after the following operations?

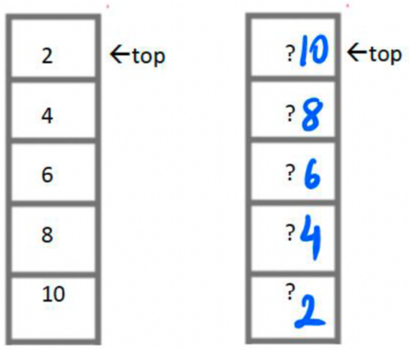
```
Queue<int> q(4);
```



```
q.enqueue(5);
q.enqueue(18);
q.enqueue(27);
q.dequeue();
q.enqueue(32);
q.dequeue();
q.enqueue(91);
```



8. Given a 5-element stack S (from top to bottom: 2, 4, 6, 8, 10), and an empty queue Q, remove the elements one-by-one from S and insert them into Q, then remove them one-by-one from Q and re-insert them into S. How does S look like (from top to bottom) ?



9. Write a template function to QueueADT class that finds the max element in the queue. You may only use queue operations such as enqueue, dequeue, size etc. No other data structure can be used other than queues. Queue must remain intact after finding the max.

```
template <class T>
virtual T QueueInterface<T>::findMax(){
   if(!isEmpty()){
      T first = peekFront();
      T max = first;
      dequeue();
      enqueue(first);
      T temp;
      do{
        temp = peekFront();
        if(temp > max)
          max = temp;
        dequeue();
        enqueue(temp);
      }while(temp != first);

      return max;
   }

}
```