CSCI 313 Data Structure

Midterm 1 - B

First Name:_____

Last Name:_____

CUNY ID:_____

1. **ArrayQueue** is defined as following(**12 points**):

```
Class ArrayQueue<E> implements Queue<E>, Iterable<E>{
    private int capacity =10;
    private E[] elements = (E[]) new Object[capacity];
    private int size;
    private int first;
    private int last;
    //methods we implemented in class.
    public Iterator<E> iterator(){
        return new QueueIterator();
    }
    //Implement its iterator class for ArrayQueue
    private class QueueIterator implements Iterator<E>{
        //you may declare variables here.

        private int index = first



        public QueueIterator(){









        }
        Public Boolean hasNext(){
            return index!= last;



        }
        Public E next(){

            E  val = elements[index++];
            If(index == capacity){
                Index = 0;
            }
            return val;
        }
        Public void remove(){
            throw new NoSupportedOperationException();
        }
    }
}
```

2. Estimate the runtime using Big-O (**8 points**):

   a) What's the runtime of **hasNext()** and **next()** method in Q1.

      O(1)

   b) $T(n) = 10\log_2(n^2) + 15\log_2((2n)^2) + 5\log_2(n^5) + 200\log_2(1000^5)^2 + 10000$

      O(lgn)

   c) $T(n) = 60T(n/2) + 10n^4$

      a = 60, b =2, d = 4

      $O(n^{\log}2^{60})$

   d) $T(n) = 9T(n/3) + 100n^2$

      $O(n^2\log_3 n)$

   e) $T(n) = 63(n/4) + 3n^3$

      $O(n^3)$

f) T(n) = 2T(n - 3) + 1 Assume n = k* 3 + 1, k is 1,2,3,4,5,6…….

$T(n-3) = 2T(n-6) + 1$
$T(n-6) = 2T(n-9) + 1$
$T(1) = 1$
$T(n) = 2T(n-3) + 1 = 2^2T(n-4) + 2 + 1$
$= 2^kT(1) + 2^{k-1} + \ldots + 2 + 1$
$= O(2^{n/3})$

3. **Remove Last element in Queue(5 points)**
Define a method to remove the last element of queue. Other elements remains the same order.
You can only invoke enqueue(), dequeue(), size(), isEmpty(), peek() methods. You should NOT declare any datastructures. 0 points if you have more than 10 lines code.
Ex1:

```
Queue<Integer> s = new Queue<>();
//push 1 2 3 4 5 into queue.
removeLast(s): return 5
removeLast(s): return 4
removeLast(s): return 3
```

```
public E removeLast(Queue<E> queue){
    for(int i= 1;i<size;i++){
        queue.enqueue(queue.dequeue());
    }
    return queue.dequeue();




}
```