CSc 115 Fundamentals of Programming II

Pseudo Quiz 3 February 2014 **NAME:**________________________________

1) An algorithm's execution time is related to the number of ______ it requires.
   - a) parameters
   - b) test data sets
   - c) data fields
   - [✓ d) operations]

2) The efficiency of an algorithm's execution is typically a concern for ______.
   - a) small problems only
   - [✓ b) large problems only]
   - c) medium sized problems only
   - d) problems of all sizes

3) If a problem of size n requires execution time that is directly proportional to n, the problem is said to be ______.
   - a) O(1)
   - [✓ b) O(n)]
   - c) $O(n^2)$
   - d) O(2n)

4) A growth-rate function of ______ implies a problem whose execution time requirement is constant.
   - [✓ a) O(1)]
   - b) O(n)
   - c) $O(2^n)$
   - d) $O(n^2)$

5) Consider an algorithm that contains loops of the form:

```
for (x = 1 through n ) {
  for (y = 1 through n) {
    for (z = 1 through 10) {
         Task T.
    }  // end for
  }  // end for
}  // end for
```

   If task T requires a constant amount of execution time, the execution time for the entire algorithm is ______.
   - a) O(1)
   - b) O(n)
   - [✓ c) $O(n^2)$]
   - d) O(10n)
   - e) $O(10n^2)$

6) Assume a linked list contains n nodes and consider the following the code fragment:

```
Node curr = head;
while (curr != null) {
      System.out.println(curr.getItem());
      curr = curr.getNext();
}  // end while
```

   The above code requires ______ time for execution.
   - a) O(1)
   - [✓ b) O(n)]
   - c) $O(n^2)$
   - d) O(log n)

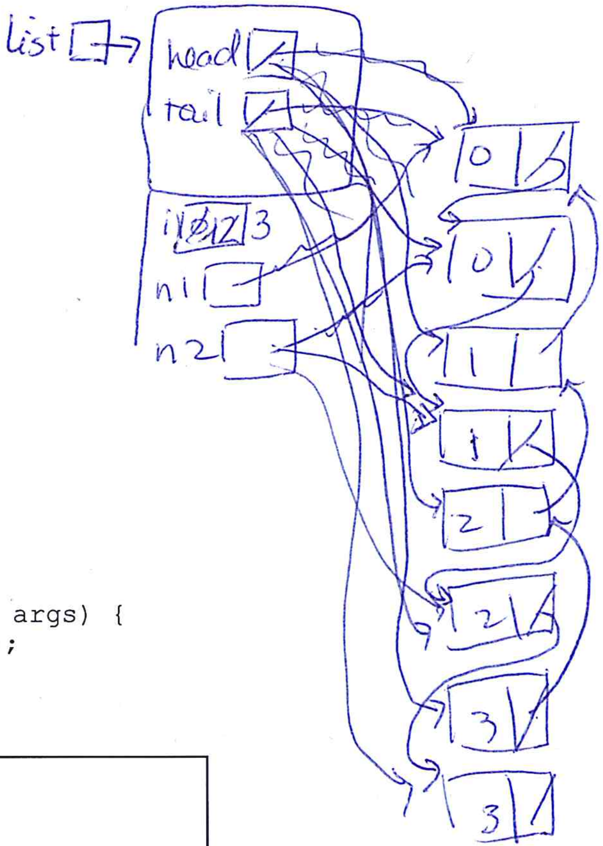
7) Determine the output created by the following code and show the resulting list that it creates, i.e., draw the "boxes and arrows". (Uses the Node class below.)

```
public class MTList {
      private    Node head;
      private Node tail;

      public MTList () {
            head = null;
            tail = null;
      }
      public MTList (int n) {
            head = null; tail = null;
            for (int i=0;i<n;i++) {
                  Node n1 = new Node();
                  n1.element = i;
                  n1.next = head;
                  head = n1;
                  if (tail == null)
                        tail = n1;
                  Node n2 = new Node();
                  n2.element = i;
                  n2.next = null;
                  tail.next = n2;
                  tail = n2;
            }
      }
      public String toString () {
            Node p = head;
            String s = "[";

            while (p != null) {
                  s += p.element;
                  if (p != tail )
                        s+= ",";
                  p = p.next;
            }
            s += "]";
            return s;
      }

      public static void main (String[] args) {
            MTList  list = new MTList(4);
            System.out.println(list);
      }
}
```

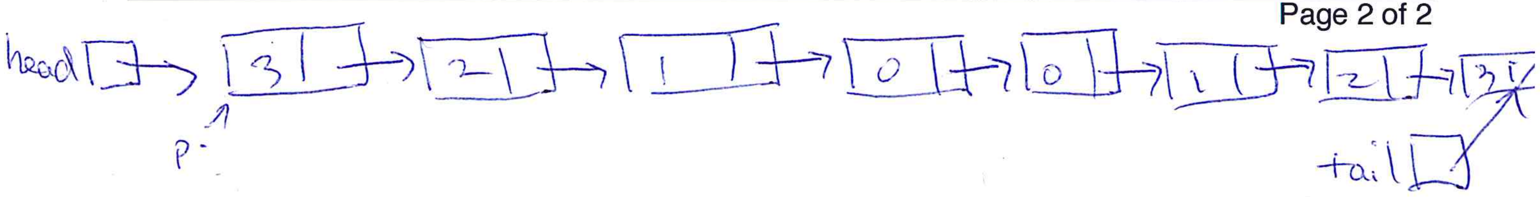
Output:

[3,2,1,0,0,1,2,3]

Draw the list :



```
public class Node  {
      public double element;
      public Node next;

      public Node() {
            element = null;
            next = null;
      }
      // No getters or setters: attributes are public!
}
```



head
3
2
1
0
0
1
2
3
p
tail