1 Exceptions (Spring 2016 MT2 Q3)

Consider the code below. Recall that x / 2 rounds down to the nearest integer.

```java
public static void checkIfZero(int x) throws Exception {
    if (x == 0) {
        throw new Exception("x was zero!");
    }
    System.out.println(x); // PRINT STATEMENT
}
public static int mystery(int x) {
    int counter = 0;
    try {
        while (true) {
            x = x / 2;
            checkIfZero(x);
            counter += 1;
            System.out.println("counter is " + counter); // PRINT STATEMENT
        }
    } catch(Exception e) {
        return counter;
    }
}
public static void main(String[] args) {
    System.out.println("mystery of 1 is " + mystery(1));
    System.out.println("mystery of 6 is " + mystery(6));
}
```

What will be the output when main is run?

```
mystery of 1 is 0

3
counter is 1

1
counter is 2
mystery of 6 is 2
```

2   AltList (Summer 2016 MT2 Q2)

A normal generic linked list contains objects of only one type. But we can imagine a generic linked list where entries alternate between two types. `AltList` is an implementation of such a data structure:

```
1  public class AltList<X, Y> {
2      private X item;
3      private AltList<Y, X> next;
4
5      AltList(X item, AltList<Y, X> next) {
6          this.item = item;
7          this.next = next;
8      }
9  }
```

Let's construct an AltList instance:

```
1  AltList<Integer, String> list =
2      new AltList<Integer, String>(5,
3          new AltList<String, Integer>("cat",
4              new AltList<Integer, String>(10,
5                  new AltList<String, Integer>("dog", null))));
```

This list represents [5 cat 10 dog]. In this list, assuming indexing begins at 0, all even-index items are `Integer`s and all odd-index items are `String`s.

Write an instance method called `pairsSwapped()` for the `AltList` class that returns a copy of the original list, but with adjacent pairs swapped. Each item should only be swapped once. This method should be non-destructive: it should not modify the original `AltList` instance.

For example, calling `list.pairsSwapped()` should yield the list [cat 5 dog 10]. There were two swaps: "cat" and 5 were swapped, then "dog" and 10 were swapped. You may assume that the list on which `pairsSwapped()` is called has an **even non-zero** length. Your code should maintain this invariant.

```
1  public class AltList<X, Y> {
2      public ____AltList <Y, X>____ pairsSwapped() {
3
4          if ( next.next == null){
5
6              return new AltList<Y,X>(next.item, new AltList<X,Y> (item, null ));
7
8          }
9          return new AltList<Y, X> (next.item ,
10
11                 new AltList <X,Y> (item, next.next.pairsSwapped ));
12
13      }
14  }
```

3 Every κth Element (Fall 2014 MT1 Q5)

Fill in the `next()` method in the following class. Do not modify anything outside of `next`.

```
1   import java.util.Iterator;
2   import java.util.NoSuchElementException;
3   /** Iterates over every Kth element of the IntList given to the constructor.
4    *   For example, if L is an IntList containing elements
5    *   [0, 1, 2, 3, 4, 5, 6, 7] with K = 2, then
6    *       for (Iterator<Integer> p = new KthIntList(L, 2); p.hasNext(); ) {
7    *           System.out.println(p.next());
8    *       }
9    *   would print get 0, 2, 4, 6. */
10  public class KthIntList implements Iterator <Integer> {
11      public int k;
12      private IntList curList;
13      private boolean hasNext;
14
15      public KthIntList(IntList I, int k) {
16          this.k = k;
17          this.curList = I;
18          this.hasNext = true;
19      }
20
21      /** Returns true iff there is a next Kth element. Do not modify. */
22      public boolean hasNext() {
23          return this.hasNext;
24      }
25
26      /** Returns the next Kth element of the IntList given in the constructor.
27       *  Returns the 0th element first. Throws a NoSuchElementException if
28       *  there are no Integers available to return. */
29      public Integer next() {
30          if (curList.isEmpty()) {
31              throw new NoSuchElementException();
32          }
33          int result = curList.removeFirst();
34          if (curList.size() < k){ hasNext = False;}
35          for (int i=0; i<k-1; i++){
36              curList.removeFirst();
37          }
38          return result;
39
40      }
41  }
```