

1 Exceptions (Spring 2016 MT2 Q3)

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

```
1 public static void checkIfZero(int x) throws Exception {
2     if (x == 0) {
3         throw new Exception("x was zero!");
4     }
5     System.out.println(x); // PRINT STATEMENT
6 }
7 public static int mystery(int x) {
8     int counter = 0;
9     try {
10        while (true) {
11            x = x / 2;
12            checkIfZero(x);
13            counter += 1;
14            System.out.println("counter is " + counter); // PRINT STATEMENT
15        }
16    } catch (Exception e) {
17        return counter;
18    }
19 }
20 public static void main(String[] args) {
21     System.out.println("mystery of 1 is " + mystery(1));
22     System.out.println("mystery of 6 is " + mystery(6));
23 }
```

What will be the output when main is run?

1 _____
2 _____
3 _____
4 _____
5 _____
6 _____
7 _____
8 _____
9 _____
10 _____
11 _____
12 _____

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 `Integers` and all odd-index items are `Strings`.

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 _____ pairsSwapped() {
3
4         _____
5
6         _____
7
8         _____
9
10        _____
11
12        _____
13    }
14 }

```

3 Every k th 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         -----
31         -----
32         -----
33         -----
34         -----
35         -----
36         -----
37         -----
38         -----
39         -----
40     }
41 }

```