Lab 12 Notes

CS 61BL
Summer 2019

Lab 12
July 18, 2019

Name:

SID:

Please complete this worksheet during your lab, and turn it in to your TA by the end of your section. You are encouraged to work with your neighbors collaboratively.

Section Number: ○ 01 ○ 02 ○ 03 ○ 04 ○ 05 ○ 06 ○ 07 ○ 08 ○ 09 ○ 10 ○ 11 ○ 12

1 Exceptions

1.1 Consider the code below. Trace through the main method below and determine what Java would display. (Your solution should use all 6 blanks.)

Hint: You can check your solution by pasting the code into the online Java visualizer.

```java
public class ExceptionsPuzzle {
    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));
    }
} // continued on next page
```

1/2 = 0 (floor divide)

mystery of 1 is 0
3
Counter is 1
1
Counter is 2
mystery of 6 is 2

Write what would be printed below.

1. ______________________________

2. ______________________________

3. ______________________________

4. ______________________________

5. ______________________________

6. ______________________________

2 Iterators

2.1 Consider the following code, intended to be included in the `SpaceList` class.

For each code snippet, determine whether or not the code snippet will perform correctly according to the requirements of the Iterator interface. Suppose we always start with an `SpaceList<Integer>` list containing just the number 5.

```java
private class BadSpaceListIterator implements Iterator<Item> {
    private int bookmark = 0;
    private boolean done = false;
    public boolean hasNext() {
        if (done) {
            return false;
        }
        if (bookmark == size - 1) {
            done = true;
        }
        return true;
    }
    public Item next() {
        Item rtn = values[bookmark];
        bookmark += 1;
        return rtn;
    }
} // continued on next page
```

```
Iterator<Integer> iter = list.iterator();
boolean b;
b = iter.hasNext();  true
b = iter.hasNext();  false
```

○ Performs correctly

● Does not perform correctly

```
Iterator<Integer> iter = list.iterator();
                      true
if(iter.hasNext()) {
    System.out.println(iter.next());  good
}
                 true X
if(iter.hasNext()) {
    System.out.println(iter.next());  ← fail
}
```

○ Performs correctly

● Does not perform correctly

```
Iterator<Integer> iter = list.iterator();
System.out.println(iter.next());
System.out.println(iter.next());
```

← undefined behavior (array out of bounds? garbage?)

code itself is just bad, user error not iterator error

- just don't do this.

● Performs correctly

○ Does not perform correctly

```
Iterator<Integer> iter = list.iterator();
boolean b;
int n;
n = iter.next();  good
b = iter.hasNext();  true X
```

○ Performs correctly

● Does not perform correctly

2.2 Fill in the following code, intended to be included in the SLList class, such that the Iterator below returns every nth element in the SLList (e.g. a SLListSkipIterator with input 2 returns every other item).

```
private class SLListSkipIterator implements Iterator<Item> {

    SLListSkipIterator(int n) {
        bookmark = sentinel.next
        skip = n
    }

    private ListNode<Item> bookmark;
    private int skip;

    public Item next() {
        Item toReturn = bookmark.item
        for (int i = 0; i < skip; i++) {
            bookmark = bookmark.next
            if (bookmark = sentinel) {
                skip = -1
            }
        }
        return toReturn;
    }

    public boolean hasNext() {
        return skip != -1
    }
}
```