[3, 4]

This list of integers makes BinarySearch throw a Stackoverflow error when you search for anything less than the smallest key in that list (e.g. 2, 1, 0) because it goes into an infinite loop. What happens is that the middle index gets assigned to 0 ([0+1]/2=0) which is in our case the number 3. 3 is larger than let's say 2 (the key being searched), so diff is negative. In that case beginIndex stays constant; however endIndex is assigned to middleIndex-1 which is -1 to look for the key in the lower half. From there onwards, middleIndex gets assigned to 0 ((-1 + 0) / 2) again and again and the program keeps comparing the first number with the number you are searching and fails to do any progress. It gets stuck in an infinite loop. Because recursion is implemented using Stacks, an infinite loop throws a Stackoverflow error. This is why we need to check the following before running another search function:

```
if (beginIndex > endIndex)
  return -1;
```

If you search for a key that is strictly greater than every element in this list, we don't get the error because

```
if (beginIndex == endIndex)
                    return key.equals(list.get(beginIndex)) ? beginIndex : -1

this piece returns -1.
```

Other possible lists: [3, 3, 3, 4, 4, 4] – [3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4] …

*Numbers are arbitrary; size is critical and the key you search for must be less than any element in the list.