

int[]

int[]

Write a static method named `insertMiddle` that accepts two arrays of integers `a` and `b` as parameters and returns a new array containing elements from the first half of `a` followed by all the elements of `b` followed by elements from the second half of `a`. For example, consider the following two arrays:

```
int[] a = {2, 4, 6, 8, 10};
int[] b = {1, 1, 1};
```

traversal → for loops?

The call `insertMiddle(a, b);` should return the following array:

```
{2, 4, 1, 1, 1, 6, 8, 10}
```

special case (*)
a odd length

Notice that if `a` has an odd length, its shorter half goes first.

You may not construct any extra data structures or String objects to solve this problem. You may not modify the arrays that are passed in.

no extra arrays
no strings

Steps:

- create result array
- copy first half of `a`
- copy `b`
- copy second half of `a`

```
public static int[] insertMiddle
    (int[] a, int[] b) {
    int[] result = new int[a.length + b.length];
    for (int i = 0; i < a.length / 2; i++) {
        result[i] = a[i];
    }
    for (int i = 0; i < b.length; i++) {
        result[a.length / 2 + i] = b[i];
    }
    for (int i = a.length / 2; i < a.length; i++) {
        result[b.length + i] = a[i];
    }
    return result;
}
```

Write a method flip that takes a Random object as a parameter and that prints information about a coin-flipping simulation.

Your method should use the Random object to produce a sequence of simulated coin flips, printing whether each flip comes up "heads" or "tails". Each outcome should be equally likely. Your method should stop flipping when you see three heads in a row. It should return the total number of flips.

2 options, nextInt(2)

int

counter

For example, if we construct a Random object and make the following calls:

```
Random r = new Random();
flip(r);
flip(r);
```

while loop counter

We expect to get a log of execution like this:

development strategy

```
heads
tails
heads
heads
heads
3 heads in a row after 5 flips
```

- random flip + print
- while loop + counter
- heads in a row
- total flips counter
- return

```
heads
heads
tails
heads
tails
tails
heads
heads
heads
3 heads in a row after 9 flips
```

You must exactly reproduce the format of the log above.

note: need to start over counting heads in a row if we get tails!

```
public static int flip(Random r) {
    int headsInRow = 0;
    int count = 0;
    while (headsInRow < 3) {
        int next = r.nextInt(2); // 0 or 1
        if (next == 0) { // heads
            headsInRow++;
            s.o.println("heads");
        } else { // tails
            headsInRow = 0;
            s.o.println("tails");
        }
    }
}
```

count++;

s.o.println("3 heads in a row after" + count + "flips");
s.o.println();

maximum

Scanner

Write a static method named longestName that reads names typed by the user and prints the longest name (the name that contains the most characters) in the format shown below. Your method should accept a console Scanner and an integer n as parameters and should then prompt for n names.

The longest name should be printed with its first letter capitalized and all subsequent letters in lowercase, regardless of the capitalization the user used when typing in the name.

If there is a tie for longest between two or more names, use the tied name that was typed earliest. Also print a message saying that there was a tie, as in the right log below. It's possible that some shorter names will tie in length, such as DANE and Erik in the left log below; but don't print a message unless the tie is between the longest names.

You may assume that n is at least 1, that each name is at least 1 character long, and that the user will type single-word names consisting of only letters. Two sample calls and their output are shown below.

```
Scanner console = new Scanner(System.in);
longestName(console, 5);
```

```
name #1? roy
name #2? DANE
name #3? Erik
name #4? sTeFaNiE
name #5? Laura
Stefanie's name is longest
```

```
Scanner console = new Scanner(System.in);
longestName(console, 7);
```

```
name #1? TrEnt
name #2? rita
name #3? JORDAN
name #4? craig
name #5? leslie
name #6? YUKI
name #7? TaNnEr
Jordan's name is longest
(There was a tie!)
```

```
public static void longestName(Scanner s,
    String longest = "";
    boolean tie = false;
    for(int i=1; i<=n; i++){
        S.o.p("name #" + i + "? ");
        String name = s.next();
        if(name.length() > longest.length()){
            longest = name;
            tie = false;
        } else if(name.length() == longest.length()){
            tie = true;
        }
    }
}
```

next()
development strategy
o prompt & read n names
o longest
o tie
o formatting

if have a new longest name, notice for that length yet

edge case

str methods
to up...
to low...

```
longest = longest.toUpperCase().charAt(0) +
    longest.toLowerCase().substring(1);
S.o.p(longest + "'s name was longest");
```

```
if(tie){
    S.o.pn("(There was a tie!)");
}
```

str methods
toLp...
toLow...

Write a static method called `hashTag` that takes a `String` as a parameter and returns the `String` converted to a hashtag. A hashtag is a `String` made up of a pound sign (`#`) followed by a phrase, where each word is capitalized (with the first letter uppercase, and the other letters lowercase). Words in the original string are separated by one or more spaces. For example, the call `hashTag("I love computer science")` would return `"#ILoveComputerScience"`

Below are more sample calls:

Method Calls	Value Returned
<code>hashTag("I love computer science")</code>	<code>"#ILoveComputerScience"</code>
<code>hashTag("to be or not to be")</code>	<code>"#ToBeOrNotToBe"</code>
<code>hashTag("saY YES")</code>	<code>"#SayYes"</code>
<code>hashTag(" edGAR allan pOE ")</code>	<code>"#EdgarAllanPoe"</code>
<code>hashTag(" sPoo000oo0o0ky")</code>	<code>"#Spoooooooooooky"</code>
<code>hashtag(" fuNNY #@*^!& sYMBOLs ")</code>	<code>"#Funny#@*^!&Symbols"</code>
<code>hashTag("x")</code>	<code>"#X"</code>
<code>hashTag(" ")</code>	<code>"#"</code>
<code>hashTag("")</code>	<code>"#"</code>

traversal to find where words begin/end?
Cumul. algo. to build up str?

Notice that words can contain other punctuation. Any non-empty sequence of non-space characters can be a word. Also notice that there may be spaces at the beginning or end of the `String`. You may not construct any `Scanners` or `tokenizers` to solve this problem. You may assume that the `String` has no other whitespace characters such as tabs or newline characters.

edge cases

*

In addition to the `String` methods on your cheat sheet, you may use the methods `Character.toUpperCase()` and `Character.toLowerCase()`, which take a `char` as a parameter and returns the `char` in uppercase and lowercase, respectively. For example, `Character.toUpperCase('a')` returns `'A'`. If the character is already in uppercase or is not a letter, it returns the character unchanged (e.g., `Character.toUpperCase('A')` returns `'A'` and `Character.toUpperCase('@')` returns `'@'`).

```
public static String hashTag(String s){
```

```
String result = "";
boolean firstLetter = true;
for (int i=0; i<s.length(); i++){
    if (s.charAt(i) == ' '){
```

if this is a space, next char will be the first char of new word

```
        firstLetter = true;
```

```
    } else {
        if (firstLetter){
            result += s.toUpperCase().charAt(i);
            firstLetter = false;
```

if we just processed 1st letter of a word, the next one isn't the first letter anymore

```
    } else {
        result += s.toLowerCase().charAt(i);
```

```
    }
}
return "#" + result;
```