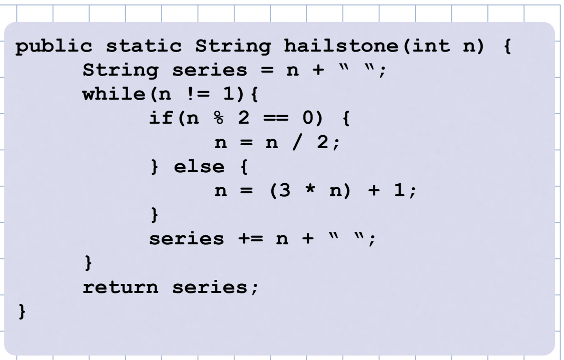
hailstone(5)



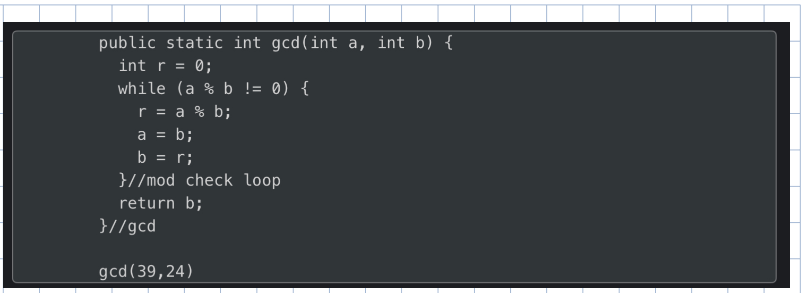
```
public static String hailstone(int n) {
      String series = n + " ";
      while(n != 1){
            if(n % 2 == 0) {
                  n = n / 2;
            } else {
                  n = (3 * n) + 1;
            }
            series += n + " ";
      }
      return series;
}
```

Table trace diagram
variables: n, series   (n is int, series string)

Loop #	Variable	Value
—	n series	5 "5 "
1	n series	16 "5 16 "
2	n series	8 "5 16 8 "
3	n series	4 "5 16 8 4 "
4	n series	2 "5 16 8 4 2 "
5	n series	1 "5 16 8 4 2 1 "

gcd(39,24)

Variables:
a
b
r
} all ints



```
public static int gcd(int a, int b) {
  int r = 0;
  while (a % b != 0) {
    r = a % b;
    a = b;
    b = r;
  }//mod check loop
  return b;
}//gcd

gcd(39,24)
```

Loop #	a	b	r	Work it out
—	39	24	0	(parameters, declaration & initialization of r)

1	24	15	15	a % b != 0 TRUE
				r = a % b = 39 % 24 = 15
				a = b = 24
				b = r = 15

2	15	9	9	a % b != 0 TRUE
				r = a % b = 24 % 15 = 9
				a = b = 15
				b = r = 9

3	9	6	6	a % b != 0 TRUE
				r = a % b = 15 % 9 = 6
				a = b = 9
				b = r = 6

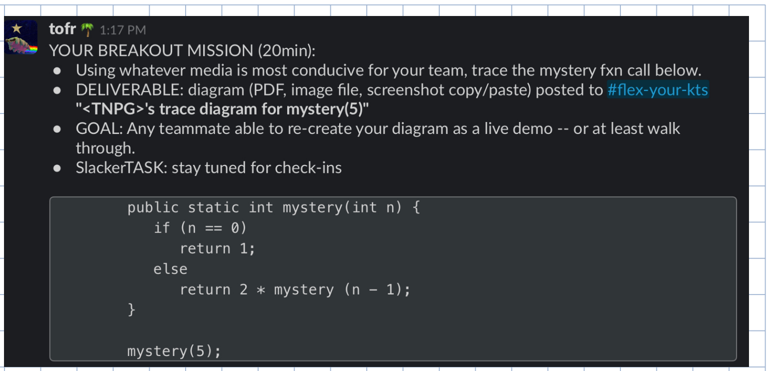
4	6	3	3	a % b != 0 TRUE
				r = a % b = 9 % 6 = 3
				a = b = 6
				b = r = 3

5				a % b != 0 FALSE ↳ return b → return 3

Trace diagram

Variable: n

Working on:
mystery(5);



tofr 🌴 1:17 PM

YOUR BREAKOUT MISSION (20min):

- Using whatever media is most conducive for your team, trace the mystery fxn call below.
- DELIVERABLE: diagram (PDF, image file, screenshot copy/paste) posted to #flex-your-kts **"<TNPG>'s trace diagram for mystery(5)"**
- GOAL: Any teammate able to re-create your diagram as a live demo -- or at least walk through.
- SlackerTASK: stay tuned for check-ins

```
public static int mystery(int n) {
    if (n == 0)
        return 1;
    else
        return 2 * mystery (n - 1);
}

mystery(5);
```

Loop #	n	Work it out
—	5	parameter
1	4	n == 0 FALSE return 2 * mystery(n-1) → return 2* mystery(4) = 2*16 = 32
2	3	n == 0 FALSE return 2 * mystery(n-1) → return 2* mystery(3) = 2*8 = 16
3	2	n==0 FALSE return 2* mystery(n-1) → return 2* mystery(2) = 2*4 = 8
4	1	n==0 FALSE return 2* mystery(n-1) → return 2* mystery(1) = 2*2 = 4
5	0	n==0 FALSE return 2* mystery(n-1) → return 2* mystery(0) = 2*1 = 2
6		n==0 TRUE ↳ return 1

mystery(5) ← returns 2 * ~~mystery(4)~~ 16 = 2 * 16 = $\boxed{32}$

mystery(4) ← returns 2 * ~~mystery(3)~~ 8 = 2 * 8 = 16

mystery(3) ← returns 2 * ~~mystery(2)~~ 4 = 2 * 4 = 8

mystery(2) ← returns 2 * ~~mystery(1)~~ 2 = 2 * 2 = 4

mystery(1) ← returns 2 * ~~mystery(0)~~ 1 = 2 * 1 = 2

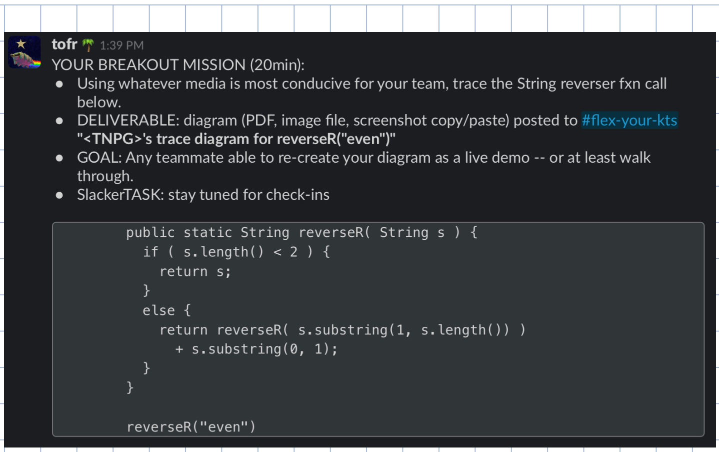
mystery(0) ← returns 1

note: mystery(int x)
equivalent to
powOf2(int x)
(will find
2^x)

reverseR

reverses String s

working on: reverseR("even")



tofr 1:39 PM

YOUR BREAKOUT MISSION (20min):

- Using whatever media is most conducive for your team, trace the String reverser fxn call below.
- DELIVERABLE: diagram (PDF, image file, screenshot copy/paste) posted to #flex-your-kts **"<TNPG>'s trace diagram for reverseR("even")"**
- GOAL: Any teammate able to re-create your diagram as a live demo -- or at least walk through.
- SlackerTASK: stay tuned for check-ins

```
public static String reverseR( String s ) {
  if ( s.length() < 2 ) {
    return s;
  }
  else {
    return reverseR( s.substring(1, s.length()) )
      + s.substring(0, 1);
  }
}

reverseR("even")
```

"nev"

reverseR("even") → return ~~reverseR("ven")~~ + "e" = "nev" + "e" = "neve"

"ne"

reverseR("ven") → return ~~reverseR("en")~~ + "v" = "ne" + "v" = "nev"

"n"

reverseR("en") → return ~~reverseR("n")~~ + "e" = "n" + "e" = "ne"

reverseR("n") → return "n"