Challenge 1: Using only loops and the methods on PriorityQueue, implement a sorting algorithm that has O(n * lg(n)) performance.

```
void pqSort(int[] arr) {
  PriorityQueue<               > pq = new PriorityQueue<>(Integer::compare);









}
```

Challenge 2

- `poll()` removes and returns the max/min element from a PriorityQueue
- `peek()` returns (without removing) the max/min element from a PriorityQueue
- Using `Integer::compare` as the comparator for Java's default PQ makes a MIN heap

First, try describing what the code does in your own words. Consider adding a sequence of numbers and thinking about pq1, pq2!

```
class ____________Tracker {
  PriorityQueue<Integer> pq1 = new PriorityQueue<>(Collections.reverseOrder(Integer::compare));
  PriorityQueue<Integer> pq2 = new PriorityQueue<>(Integer::compare);
  void add(int n) {
    if(pq2.size() == 0 && pq1.size() == 0) {
      pq2.add(n);
      return;
    }
    int current = get();
    if(n >= current) {
      pq2.add(n);
    }
    else {
      pq1.add(n);
    }
    int sizeDifference = pq2.size() - pq1.size();
    if(sizeDifference > 1) { pq1.add(pq2.poll()); }
    else if(sizeDifference < -1) { pq2.add(pq1.poll()); }
  }

  int get() {
    if(pq2.size() == pq1.size()) { return (pq2.peek() + pq1.peek()) / 2; }
    if(pq2.size() > pq1.size()) { return pq2.peek(); }
    else { return pq1.peek(); }
  }

  public String toString() {
    return "" + pq1 + " " + this.get() + " " + pq2;
  }
}
```

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct Point {
  int x, y;
} Point;
Point* make_point(int x, int y) {
  Point* p = calloc(1, sizeof(Point));
  p->x = x;
  p->y = y;
  return p;
}

int max(int n, int m) {
  int compare_result = n > m;
  printf("Compare result for %d > %d: %d\n",
    n, m, compare_result);
  if(compare_result) { return n; }
  else { return m; }
}

int between(float low, float n, float high) {
  return n > low && n < high;
}

int sum(int upto) {
  int result = 0;
  for(int i = 0; i < upto; i += 1) {
    result += i;
  }
  return result;
}

int sumarr(int* arr, int length) {
  int result = 0;
  for(int i = 0; i < length; i += 1) {
    result += arr[i];
  }
  return result;
}

int isLeftOf(Point* p1, Point* p2) {
  return p1->x < p2->x;
}

int main(int argc, char** args) {
  printf("Hello\n");
  printf("max(4, 5): %d\n", max(4, 5));
  printf("max(6, 5): %d\n", max(6, 5));
  printf("sum to 10: %d\n", sum(10));

  int* a = calloc(5, sizeof(int));
  a[0] = 10;
  a[1] = 20;
  a[2] = 30;
  a[3] = 40;
  a[4] = 50;

  printf("sum of a: %d\n", sumarr(a, 5));

  Point* p45 = make_point(4, 5);
  Point* p78 = make_point(7, 8);

  printf("%d\n", isLeftOf(p45, p78));
} }
```

```
❱ gcc main.c -o main_to_run
❱ ./main_to_run
Hello
Compare result for 4 > 5: 0
max(4, 5): 5
Compare result for 6 > 5: 1
max(6, 5): 6
sum to 10: 45
sum of a: 150
1
```

```java
class Point {
  int x, y;
  Point(int x, int y) {
    this.x = x;
    this.y = y;
  }
}
class Main {


  static int max(int n, int m) {
    boolean compareResult = n > m;
    System.out.print("Compare result for " + n + " > " + m +
": " + compareResult + "\n");
    if(compareResult) { return n; }
    else { return m; }
  }

  static boolean between(float low, float n, float high) {
    return n > low && n < high;
  }

  static int sum(int upto) {
    int result = 0;
    for(int i = 0; i < upto; i += 1) {
      result += i;
    }
    return result;
  }

  static int sumarr(int[] arr) {
    int result = 0;
    for(int i = 0; i < arr.length; i += 1) {
      result += arr[i];
    }
    return result;
  }

  static boolean isLeftOf(Point p1, Point p2) {
    return p1.x < p2.x;
  }

  public static void main(String[] args) {
    System.out.println("Hello");
    System.out.println("max(4, 5): " + max(4, 5));
    System.out.println("max(6, 5): " + max(6, 5));
    System.out.println("sum to 10: " + sum(10));

    int[] a = new int[5];
    a[0] = 10;
    a[1] = 20;
    a[2] = 30;
    a[3] = 40;
    a[4] = 50;

    System.out.println("sum of a: " + sumarr(a));

    Point p45 = new Point(4, 5);
    Point p78 = new Point(7, 8);

    System.out.println(isLeftOf(p45, p78));
  } }
```

```
❱ javac Main.java
❱ java Main
Hello
Compare result for 4 > 5: false
max(4, 5): 5
Compare result for 6 > 5: true
max(6, 5): 6
sum to 10: 45
sum of a: 150
true
```

Consider calling add with values 1, 7, 5, 10, 3. After:

What will the **size** of pq1, pq2 be?

A:1, 4 B: 5, 5 C: 2, 3 D: 3, 2 E: 3, 3

What will be the result of get()?

A: 1 B: 3 C: 5 D: 10 E: 3