Q3: Debugging

The hospitals using your algorithm from "Prioritizing Patients" listened to your reflections: so, they're reinventing their priority score algorithm to use different data.

Consider a static method called `priorityScore` that computes and returns an `int` representing a patient's priority score. To do so, this method takes in four parameters:

- `int age`: the patient's age, guaranteed to be between `1-99` (inclusive)
- `int painLevel`: the patient's pain level, guaranteed to be from 1-10 (inclusive)
- `double temperature`: the patient's temperature in Fahrenheit
- `int[] familyScores`: the priority scores of all family members we have on file (note: you may assume that this array is not null and has at least one item)

Similar to P2, each patient starts out with a base score of 100, which is then increased or decreased based on the following (new) algorithm:

1. if the age of the patient is a child (younger than 12 years old) or a senior citizen (greater than 75 years old), add `20` to their score
2. if the patient's reported pain is less than `7`, add double their pain level to the score; if it is greater than or equal to `7`, add their pain level plus `10` to the score
3. if the patient's temperature is below `90.5` or above `99.5` degrees, add `5` to the score
4. after calculating their individual score, look at their family's scores:
    a. for <u>each</u> family member with a score of at least `130`, add `10` to the score.
    b. if the <u>average</u> score across <u>all</u> family members is greater than `120`, add `20` to the score. If there are no family members on file, treat the average as `0`.
5. finally, `priorityScore` should return the patient's score (<u>after</u> step 4).

This table shows the expected output for a <u>correct</u> implementation of `priorityScore`:

`Method Call`	`int[] arr value`	`Returns`
`priorityScore(21, 6, 97.3, arr);`	`{100}`	`112`
`priorityScore(21, 6, 97.3, arr);`	`{100, 140}`	`122`
`priorityScore(21, 6, 97.3, arr);`	`{100, 140, 160}`	`152`

Consider the following proposed buggy implementation of `priorityScore`. This implementation contains three bugs that are causing it to not work as intended!

Your task: Annotate (write on) the code below to indicate how you would fix the three bugs. You may add (using arrows to indicate where to insert), remove (by crossing out), or modify (with a combination) any code you choose. Each fix is "local" (i.e. it should not involve adding, removing, or changing more than one line).

```
1   public static int priorityScore(int age, int painLevel,
2                       double temperature, int[] familyScores) {
3       int score = 100;
4
5       if (age < 12 && age > 75) {
6           score += 20;
7       } else if (painLevel < 7) {
8           score += 2 * painLevel;
9       } else {
10          score += painLevel + 10;
11      }
12      if (temperature < 90.5 || temperature > 99.5) {
13          score += 5;
14      }
15
16      double avg = 0.0;
17      for (int i = 0; i < familyScores.length(); i++) {
18          avg += familyScores[i];
19          if (familyScores[i] >= 130) {
20              score += 10;
21          }
22      }
23
24      if (avg / familyScores.length > 120.0) {
25          score += 20;
26      }
27
28      return score;
29  }
```

6. Array Programming

Write a static method called **`sumArrays`** that sums the elements from a pair of arrays. Your method should take two parameters:

- `int[] nums1` - the first array of integers to consider
- `int[] nums2` - the second array of integers to consider

Your method should *return* a new array where each element of the new array contains the sum of the two integers at that index in the parameter arrays. If one array is longer than the other, the result array should contain the extra numbers from the longer array.

For example, suppose the following code was executed:

```
int[] nums1 = {1, 2, 3, 4, 5};
int[] nums2 = {10, 20, 30, 40, 50};
int[] result = sumArrays(nums1, nums2);
```

After this code is executed, the array result would contain the following values:

```
[11, 22, 33, 44, 55]
```

Note that the sum of the integers at each index was put into the result array (for instance 1 plus 10 is 11 and 30 plus 3 is 33).

As another example, suppose the following code was executed:

```
int[] nums1 = {1, 2, 3, 4, 5, 6, 7};
int[] nums2 = {10, 20, 30, 40, 50};
int[] result = sumArrays(nums1, nums2);
```

In this case, after this code is executed, `result` would contain the following values:

```
[11, 22, 33, 44, 55, 6, 7]
```

Notice that the extra elements 6 and 7 from the longer parameter array were also included in the result, but without being summed with anything.

You may assume that both parameter arrays contain at least one element.

Write your solution in the box on the next page.

Write your solution to problem #6 here: