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: this array may be empty, in the case where we don't have their family records)

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 each family member with a score of at least `130`, add `10` to the score.
    b. if the average score across all 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 (after step 4).

This table shows the expected output for a correct implementation of `priorityScore`:

`Method Call`	`int[] arr value`	`Returns`
`priorityScore(21, 6, 97.3, arr);`	`{}`	`112`
`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 four 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 require significant work).

```
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     int avg = 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 (1.0 * avg / familyScores.length > 120) {
25         score += 20;
26     }
27
28     return score;
29  }
```

Here is one solution; note that it fixes five bugs (all highlighted).
Normally, on a debugging problem, there will only be 3 bugs (and 3 to get an E); I added more here to give you exposure to different types of bugs!

```
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       }
8       if (painLevel < 7) { // change from else if to if
9           score += 2 * painLevel;
10      } else {
11          score += painLevel + 10;
12      }
13      if (temperature < 90.5 || temperature > 99.5) {
14          score += 5;
15      }
16
17      int avg = 0;
18      for (int i = 0; i < familyScores.length; i++) { // no ()
19          avg += familyScores[i];
20          if (familyScores[i] >= 130) { // "at least" in spec
21              score += 10;
22          }
23      }
24
25      if (familyScores.length > 0) { // check for empty arr
26          if (1.0 * avg / familyScores.length > 120) {
27
28              score += 20;
29
30          }
31      }
32
33      return score;
34  }
```