**Findings summary**

1. App stops count rolls after 9 frames even if there are rolls left

2. App adds 2 steps to a frame instead of 1 in the "Open frame". Actually, it uses frames instead of rolls as a basic unit, which lead to confusion and was a cause of majority of the bugs that I have found.

3. App has no safeguards to prevent out of bounds exception when counting rolls if there is less than 12 rolls in total

4. During final frame, if it's “Spare”, game supposed to count last 3 rolls as one frame. Instead, in addition to a "Spare" points it counts third roll one more time.

Example:

Final frame - [5, 5, 5].

Expected result: 5 + 5 + 5 = 15.

Actual result: (5 + 5 + 5) + 5 = 20.

5. After changes made app essentially stopped working on frames and begun working on rolls, completely stopping accounting for false positive spares that appear on conjunction of frames. Fixed by reintroducing frame system in a different way.

Example:

Frame x – [1, 4]

Frame x+1 – [6, 3]

Expected result: 1 + 4 + 6 + 3 = 14

Actual result: 1 + (4 + 6) + 3 + 3 = 17

**Recommendations**

1. Give more attention to user requirements. If program were implemented according to them from the beginning, there wouldn’t be so many errors in the first place.

2. Design functions with testing and maintenance in mind, make them scalable if possible.

Program was too rigid even for it’s own purpose, crashing if there is less than 9 rolls and not counting any rolls past 9. I had do essentially rework score function to be able to test it in the first place, it’s a loss of time.

3. Do not mix code and documentation, write self-documenting code instead (i.e. easily readable, not the one that pastes documentation into itself). Documentation takes a lot of space. In presented code, there was too much comments for too small functions – it was easier to ignore and/or remove comments altogether than trying to understand code by relying on them. It’s a massive waste of time.

4. Avoid writing anything remotely serious on programming languages with weak and/on dynamic typing (aka Python, JS, etc), with them maintenance will cost much more in the long run than any upfront time investment of stricter languages like C# or Java. In such languages majority of problems can be detected even before the program is launched by simple IDE code analysers.