**Program 1:**

**What is the problem with the program?**

The game crashes when a odd number of rows and columns are selected. This requires an even number of rows or columns in at least one category.

**What solution did you implement?**

I modified the reset\_game() function to ensure that the total number of tiles is always even. This rounds the value for the columns or rows up to the nearest even number.

**Does your solution have any drawbacks or limitations? Explain why or why not.**

I suppose a drawback could be that the user is not notified that the change has occurred. A pop up message might be effective in notifying the user. There is also no limit on grid size so this may present performance issues.

**Program 2:**

**What is the problem with the program?**

The program crashes due to an index out of range error in the duplicate counting loop.

**What solution did you implement?**

Fixed the out of range error by using the manual loop with collections.counter which counts occurrences. Also used pythons built in sort instead of quicksort which increases performance.

**Does your solution have any drawbacks or limitations? Explain why or why not.**

Counter stores a dictionary of al unique values which consumes a lot of memory for very large datasets. Additionally, there is not limit on data size.

**Program 3:**

**What is the problem with the program?**

The is\_prime() function is very slow because it checks divisibility up to x-1. The loop also checked many unnecessary numbers which wasted time.

**What solution did you implement?**

Instead of looping from 2 to x-1 it only checks up to √x, significantly reducing the number of iterations. Even numbers are also skipped.

**Does your solution have any drawbacks or limitations? Explain why or why not.**

The program still uses a loop for odd numbers which is not efficient.