Systematic Debugging Cheat Sheet

1. **Define the problem**
    a. What is the expected behaviour?
    b. How does this differ from what happens?

2. **Define the search space**
    a. Latest point in execution before things have gone bad*
    b. Earliest point in execution after things have gone bad*

3. **Binary Search** if you can
    a. Inspect program state near the middle of your search space
    b. Have any relevant program invariants been violated?
        i. Yes: this is the new end of your search space, goto 3
        ii. No: this is the new start of your search space, goto 3

4. **Breadth-First Search** otherwise
    a. More like traditional debugging except that you will avoid tracing into functions. Try to narrow the search space down to:
        i. A smaller part of the current function
        ii. A method call from the current function
    b. Either option above reduces your search space
        i. If the problem is in this function, fix it
        ii. Otherwise, goto 3

**gone bad = violated a relevant program invariant.*

Other tips

- You can apply this technique to bugs you can't reproduce
    Rely more on BFS, logging, and inspection (debug in your brain)
    Ensure you have appropriate logs to narrow your search space
- Take the time to setup & learn a good debugger
- Take advantage of your knowledge of how the system works to fix issues quickly Double-check assumptions about behavior

Systematic Debugging Worksheet

Problem Notes:

	Start Time	Search Space Start	Search Space End	Notes
1				
2				
3				
4				
5				
6				
7				
8				