```
int refreshAllProcs(Set<Process> procs, int maxAttempts) {
        List<Process> destroyed = new ArrayList<>();
        if (procs != null) {
                /** non-null check on procs **/
                Iterator<Process> itA = procs.iterator();
                while (itA.hasNext()) {
                        Process p = itA.next();
                        boolean ok = false;
                        int attempt = 0;
                        while (!ok && attempt < maxAttempts) {
                                if (p != null) {
                                        /** non-null check on p **/
                                        ok = p.tryRestart();
                                }
                        }
                        if (!ok) {
                                p.destroy();
                                destroyed.add(p);
                        }
                }
        }
        Iterator<Process> itB = destroyed.iterator();
        while (itB.hasNext()) {
                Process d = itB.next();
                procs.remove(d);
        }
        return procs.size();
}
```

1. Define the kill and gen operations for your analysis
2. Based on you definitions, identify the corresponding kill and gen sets in the code
3. Report the type of analysis you are using (forward/all-paths, forward/any-path, etc) and explain why
4. Identify the solution by executing your analysis for the program.

1)

- KILL(X): Removes all facts about X from the set of facts for the current program point.
- GEN(X): Adds a fact about X to the set of facts for the current program point, indicating that X has been checked for non-nullness at this program point.

2)

procs: KILL(procs) is called at the beginning of the method. GEN(procs) is called at the first if-statement, where procs is checked for non-nullness.

destroyed: KILL(destroyed) is called at the beginning of the method. This variable is never used as a method call receiver, so GEN(destroyed) is never called.

itA: KILL(itA) is called at the beginning of the method. This variable is never checked for non-nullness before being used as a method call receiver, so GEN(itA) is never called.

itB: KILL(itB) is called after the while-loop that uses itB as a method call receiver. This variable is never checked for non-nullness before being used as a method call receiver, so GEN(itB) is never called.

p: KILL(p) is called at the end of each iteration of the while-loop that uses p as a method call receiver. GEN(p) is called at the first iteration of the while-loop, where p is checked for non-nullness.

d: KILL(d) is called after the while-loop that uses d as a method call receiver. This variable is never checked for non-nullness before being used as a method call receiver, so GEN(d) is never called.

3)

To verify the property that each reference-typed variable X in the program is either checked for non-nullness before being used as a method call receiver or never checked for non-nullness before being used, we can use a forward dataflow analysis. Specifically, i will use a forward, all-paths analysis since i want to analyze all possible paths of the program.

4)

I will use the following shorthand notation for the variables:

| | |
|---|---|
| P: procs | D: destroyed |
| A: itA | B: itB |
| P2: p | D2: d |

| IN | OUT |
|---|---|
| | P,D,A,B,P2,D2 |
| P,D,A,B,P2,D2 | D,A |
| D,A | D,A,B |
| D,A,B | D2 |
| D2 | P2,D2 |
| P2,D2 | P2,D2 |
| P2,D2 | P2,D2 |
| P2,D2 | D |
| D | D |
| D,A,B | P2,D,A,B |
| P,D,A,B,P2,D2 | P,D,A,B,P2,D2 |