Exercises, Section 5.2

1. Provide reachability conditions, infection conditions, propagation conditions, and test case values to kill mutants 2, 4, 5, and 6 in Figure 5.1.

   Solution (Instructor only):

   Δ**1** *The statement will always be* **reached**. *Since we replace one variable with another, a test will* **infect** *if the variables have different values. The infection will* **propagate** *if we skip the body of the* `if` *statement.*

   R: $True$

   I: $A \neq B$

   P: $\neg(B < A) \equiv A \geq B$

   Δ**2** *The statement will always be* **reached**. *Since we change the relational operator, a test will* **infect** *if the entire predicate gives a different result. Since the infection will force a different path, the infection will always* **propagate**.

   R: $True$

   I: $(B < A) \neq (B > A) \equiv A \neq B$

   P: $True$

   Δ**3** *The statement will always be* **reached**. *Since we replace one variable with another, a test will* **infect** *if the variables have different values. But, the value of* `A` *was assigned to* `minVal` *in the previous statement, so they will always have the same value. Therefore, the mutant is* **equivalent**. **Propagation** *is not relevant for an equivalent mutant.*

   R: $True$

   I: $A \neq minVal \equiv False \rightarrow equivalent$

   P: N/A

   Δ**4** *The statement is* **reached** *if the predicate is true. A* `Bomb()` *mutant raises an immediate runtime exception, so it always* **infects**. *Likewise,* `Bomb()` *mutants always* **propagate**.

   R: $B < A$

   I: $True$

   P: $True$

   Δ**5** *The statement is* **reached** *if the predicate is true. Since we replace one variable with another, a test will* **infect** *if the variables have different values. Since* `minVal` *has been given a different value, the infection will always* **propagate**.

   R: $B < A$

   I: $A \neq B$

P: $True$

Δ6 *The statement is* **reached** *if the predicate is true. A* **failOnZero()** *mutant raises an immediate runtime exception if the expression is zero. failonZero() mutants always* **propagate**.

R: $B < A$

I: $B \neq 0$

P: $True$

2. Answer questions (a) through (d) for the mutant in the two methods, `findVal()` and `sum()`.

(a) If possible, find a test input that does **not** reach the mutant.

Solution (Instructor only):
`findVal`*: The mutant is always reached, even if* `x = null`.

`sum`*: If* `x` *is null or the empty array, ie* `x = null` *or* `[]`*, then the mutant is never reached.*

(b) If possible, find a test input that satisfies reachability but **not infection** for the mutant.

Solution (Instructor only):
`findVal`*: Infection always occurs, even if* `x = null`*, because* `i` *always has the wrong value after initialization in the loop.*

`sum`*: Any input with all zeroes will reach but not infect. Examples are:* `x = [0]` *or* `[0, 0]`.

(c) If possible, find a test input that satisfies infection, but **not propagation** for the mutant.

Solution (Instructor only):
`findVal`*: As long as the last occurrence of* `val` *isn't at* `numbers[0]`*, the correct output is returned. Examples are:* `(numbers, val) = ([1, 1], 1)` *or* `([-1, 1], 1)` *or* `(null, 0)`.

`sum`*: Any input with nonzero entries, but with a sum of zero, is fine. Examples are:* `x = [1, -1]` *or* `[1, -3, 2]`.

(d) If possible, find a test input that kills mutant m.

Solution (Instructor only):
`findVal`*: Any input with* `val` *only in* `numbers[0]` *works. An example is:* `(numbers, val) = ([1, 0], 1)`

`sum`*: Any input with a nonzero sum works. An example is:* `x = [1, 2, 3]`