```
class Bag {
  int[] a;
  int n;

  Bag(int[] input) {
    n = input.length;
    a = new int[n];
    System.arraycopy(input, 0, a, 0, n);
  }

  int extractMin() {
    int m = Integer.MAX_VALUE;
    int mindex = 0;
    for (int i = 0; i < n; i++) {
      if (a[i] < m) {
        mindex = i;
        m = a[i];
      }
    }
    n--;
    a[mindex] = a[n];
    return m;
  }
}
```

```
(1) Bag.java:6: Warning: Possible null dereference (Null)
      n = input.length;
            ^
(2) Bag.java:15: Warning: Possible null dereference (Null)
      if (a[i] < m) {
          ^
(3) Bag.java:15: Warning: Array index possibly too large (IndexTooBig)
      if (a[i] < m) {
          ^
(4) Bag.java:21: Warning: Possible null dereference (Null)
      a[mindex] = a[n];
                  ^
(5) Bag.java:21: Warning: Possible negative array index (IndexNegative)
      a[mindex] = a[n];
                  ^
```

- For each of the five warnings above, do two things:
  - explain why ESC/Java is giving you the warning (note: Do not simply say, e.g., "`input` is possibly null", rather state the source of it's possible null-ness, e.g., "`input` is passed as a parameter to the constructor and there is no check or precondition to ensure that it has a non-null value".
  - give an appropriate ESC/Java annotation to make the warning go away.

Grading: 1 point for each answer below (the italicized material in the explanation is the important part of the explanation).

- (1) Bag.java:6
  - ESC Java gives a warning that input.length could possibly involve a null dereference because there is *no guarantee that the parameter passed to the constructor is non-null.*
  - `//@ requires input != null` A post-condition is added that requires the parameter value to be non-null.
- (2) Bag.java:15
  - ESC Java gives a warning that a[i] could possibly involve a null dereference because since ESC Java does modular checking (per-method basis), it *does not see the connection between* `a` *and the parameter constructor* `input`.
  - `/*@ non_null */ int[] a;` Declare a class invariant stating that `a` is non-null.
- (3) Bag.java:15
  - ESC Java gives a warning that the index of a[i] could possibly be too large. This is because it doesn't see (from the context of the current method only) the *connection between n and the length of the array a.* Declare a class invariant that captures this information.
  - `//@ invariant 0 <= n & n <= a.length;` Declare a class invariant stating that `n` lies between 0 and `a.length`.
- (4) Bag.java:21

  - ESC Java gives a warning that a[n] could possibly involve a null dereference for the *same reasons as in item (2)* above
  - Also, the solution applied in item (2) fixes the present problem as well.

- (5) Bag.java:21
  - ESC Java gives a warning that a[n] could possibly involve a negative index value, because *nothing constrains n to be greater than one*. Add such a constraint as a precondition to the method (or a class invariant).
  - `//@ requires n >= 1;`