Indexing Checker
This add-on for the Checker Framework adds annotations to ensure that Array accesses are in Bounds and therefore prevents Runtime Index Out Of Bound Exceptions.

How To Use
There is one annotation you need to know to use this system.
@IndexFor(Array)
When you create an int that you want to use as an index for an array a you create it as such @IndexFor(a) int index = x;

You can only access array a with @IndexFor(a) int.

If you increment the index by 1 you should check it is less than the length of the array before using it to access the array.

If you decrement the index by 1 you should check it is greater than -1, or >= 0.

If you have an unknown value you should check it against both of those constraints.

You should not divide an index by any number < 1.

The result of applying any other arithmetic arithmetic operation should be treated as unknowns are(check both bounds).

Loops:

```
// this loop works properly no warnings
for(@IndexFor("arr") int i = 0; i < arr.length; i++){
    o = arr[i];
}

// due to an improper bounds check this will issue a warning for unsafe array access
for(@IndexFor("arr") int i = 0; i<= arr.length; i++){
    o= arr[i];
}

// this works fine
for(@IndexFor("arr") int i = arr.length -1 ; i > -1 /* or >= 0*/; i--){
    o= arr[i];
}
```

Unknown index value:

```
int index = input;

o = arr[index]; // issues a warning: index could be out of bounds

// this works fine no warnings because bound were properly checked
if(index > -1 /* or >= 0*/ && index < arr.length){
    o = arr[index];
}
```

What are the types:

The following is a description of every annotated type added in this checker. It is not expected that they will need to be explicitly written by developers but they may be helpful to understand warnings and debug.

@IndexTop int i	i can be any value
@IndexHigh(a) int i	i >= 0 && i <= a.length (could be one too high)
@IndexLow(a) int i	i >= -1 && i < a.length (could be one too low)
@GTZero int i	i >= 0 (could be too high)
@LTLength(a) int i	i < a.length (could be too low)
@IndexFor(a) int i	i >= 0 && i < a.length (valid access for a)
@Zero int i	i == 0
@IndexBottom int i	i can't be anything (dead code usually)

Lattice: