In lecture we derived an algebraic specifications for the abstract data type Array. I mentioned that it’s easier to specify the ADT if we make sure that instances are initialized to some known value when they are created using the new function.

Following the example specification for Array, develop a specification for uninitialized arrays. Note that even “uninitialized” arrays still need a length, but they do not have an “initial value” that gets put into every slot of the array by default; without an “initial value” it’s of course difficult to say what the get function should produce if nothing has been assigned to a slot yet.

You will have to introduce at least one new (as in not there already!) function; you’ll also have to adjust preconditions and axioms to make things work. Write up the algebraic specification for this kind of UglyArray in the format we used in lecture and comment on the differences to Array.

adt UglyArray

use Integer, Any

defines Array<any>

**Operations**

Set: array x int 🡪 array<T>

Get: array<T> x int 🡪

New: int 🡪 Array

Length: Array🡪 int

Store: the int from “New” operation

Axioms:

Get(new(n),i) = EmptyCellException

Get(Set(a, i, t), j) = (if i = j then t, else EmptyCellException)

Length(new(n)) = n

Length(set(a, I, t))

Preconditions (the parameters you put in, not the values you get out (axioms):

New(n): n > 0

Get(a, i): 0 ≤ I < a.length && array[i] **must have been set**

Set(a, I, t): 0 ≤ I < a.length

Length(a):