Verification-Driven Program Development
Exercise Sheet, Week 3

Ștefan Ciobâcă

16.10.2024

Exercises

1. Implement a method that checks whether an array of `int`s contains duplicates. Use two `for` loops. No need to specify or prove functional correctness.
2. Make it generic. Use the `(==)` *type characteristic* for the type parameter.
3. Make it return a pair of positions where the duplicates are. Use the algebraic datatype defined below to handle the error case.

```
datatype Maybe<T> = Just(what : T) | Nothing
{
  predicate IsFailure()
  {
    this.Nothing?
  }

  function PropagateFailure() : Maybe<T>
  {
    this
  }

  function Extract() : T
    requires this.Just?
  {
    this.what
  }
}
```

4. Elephant operator: make a method that takes an integer `x` as input and returns the value `(100 / x) + 10`.
5. Create a method to swap two elements in an array.
6. Implement insertion sort:
   (a) implement a method `insert` that takes an array of integers, a position `n` into the array, assumes the range between `0` and `n - 1` is sorted, and inserts the element at position `n` into the right place.
   (b) implement a method `insertSort` to sort an array by repeat calls to `insert`.
7. Implement a method `unique` that computes the unique values in an array (returns a new array, quadratic complexity).

8. Make the previous method generic.
9. Implement a method `compact` that takes a sorted array as input and removes the duplicates (in linear time).
10. Implement counting sort.
11. Implement merge sort.
12. Implement quick sort.
13. Implement radix sort.