Invariants and Encapsulation

EECS 211

Winter 2017

A struct encapsulating a binary search tree

```
struct Tree
{
    struct Node
    {
        std::string key;
        unsigned value;
        link_t left;
        link_t right;
    };
    using link_t = std::shared_ptr<Node>;

    link_t root_;
    size_t size_;
};
```

Invariants

Invariants are facts about a data structure that must always be true (for it to work properly).

- Operations must *preserve* invariants, and
- Consequently, operations can *rely* on invariants.

The `Tree` struct has invariants

For any Tree t,

- t.size_ needs to equal the actual number of elements
- For every node n, all the keys of n.left must be less than n.key
- For every node n, all the keys of n.right must be greater than n.key

The `Tree` struct has invariants

For any Tree t,

- t.size_ needs to equal the actual number of elements
- For every node n, all the keys of n.left must be less than n.key
- For every node n, all the keys of n.right must be greater than n.key

Then:

- Operations that need to know the size can safely use t.size_.
- Operations that modify need to maintain t.size_.
- Lookup operations can rely on ordering because modification operations maintain ordering.

A struct for rational numbers

```
// A rational number num/den
struct Rational
{
        long num;
        long den;
};
```

Rational representation issues

There are some issues with representing rational numbers:

Rational representation issues

There are some issues with representing rational numbers:

- Do Rational{2, 3} and Rational{4, 6} represent the same number?

Rational representation issues

There are some issues with representing rational numbers:

- Do `Rational{2, 3}` and `Rational{4, 6}` represent the same number?
- What about `Rational{2, 3}` and `Rational{-2, -3}`?

Rational representation issues

There are some issues with representing rational numbers:

- Do `Rational{2, 3}` and `Rational{4, 6}` represent the same number?
- What about `Rational{2, 3}` and `Rational{-2, -3}`?
- What does `Rational{5, 0}` mean?

Solution: `Rational` struct invariants

For any Rational r,

- r.den $>$ `0`
- gcd(r.num, r.den) $==$ `1`

Solution: `Rational` struct invariants

For any Rational r,

- r.den $>$ `0`
- gcd(r.num, r.den) $==$ `1`

These two conditions ensure that:

- We don't have nonsense rationals like Rational{`5`, `0`}.
- Every representable rational number has exactly one representation.

– To CLion! –