CS 207 Paper Exercises 2 // Due F 3/9 in class or via email to kohler@seas

Solutions by Loren McGinnis

(1) Solution:

AF(SubChooser) $= < S >$ where:
$|S| =$ `size_`,
$S = [s_0, \ldots, s_{n-1}]$ where $\forall 0 \leq i < |S|, s_i ==$ TRUE $\iff$ `value_ & (1 << i)`
RI(SubChooser) $= 0 \leq$ `size_` < 64

(2) Solution:

```
/** Returns the value of s_{@a i}
 * @pre 0 <= @a i < |S| */
BitwiseSubsetChooser::operator[](int i) const;
```

(3) Solution:

```
BitwiseSubsetChooser * subset_; //the subset that this iterator belongs to;
uint64_t mask_; //the iterator points to s_i <==> mask_ == 1 << i
```

(4) Solution:

```
std::vector<uint64_t> value_;
int size_;
```

(5) Solution:

```
/** Erase the element pointed to by @a it
 * @pre @a it is in position i, where 0 <= i < size()
 * @param it an iterator for this vector
 * @post new size() == old size() - 1
 * @post for all 0 <= j < i, new (*this)[j] == old (*this)[j]
 * @post for all i <= j < new size(), new (*this)[j] == old (*this)[j+1]
 * @return valid iterator for position i of new (*this)
 */
iterator vector<T>::erase(iterator it);
```

(6) Solution:

```
template <typename C, typename P>
void erase_if(C &x, P pred) {
  C::iterator it = x.begin();
  while (it != x.end()) {
    if (pred(*it)) {
      it = x.erase(it);
    } else {
      ++it;
    }
  }
}
```

(7) Solution:

The assumption here is that the only non-constant work that `T::erase()` does is call `value_.erase()`, which is `O(value_.size())`, and probably `O(T.size())`, assuming such a function exists.

(8) Solution:

```
U::iterator U::erase(U::iterator it) {
  assert(it.i_ < value_.size());
  int last = value_.back();
  value_.pop_back();
  if (value_.size() > it.i_) {
    *it = last;
  }
  //else it now points past-the-end
  return it;
}
```