Table 97 — Seed sequence requirements (continued)

Expression	Return type	Pre/post-condition	Complexity
`S()`		Creates a seed sequence with the same initial state as all other default-constructed seed sequences of type `S`.	constant
`S(ib,ie)`		Creates a seed sequence having internal state that depends on some or all of the bits of the supplied sequence [`ib`, `ie`).	$\mathcal{O}$(`ie` − `ib`)
`S(il)`		Same as `S(il.begin(), il.end())`.	same as `S(il.begin(), il.end())`
`q.generate(rb,re)`	`void`	Does nothing if `rb == re`. Otherwise, fills the supplied sequence [`rb`, `re`) with 32-bit quantities that depend on the sequence supplied to the constructor and possibly also depend on the history of `generate`'s previous invocations.	$\mathcal{O}$(`re` − `rb`)
`r.size()`	`size_t`	The number of 32-bit units that would be copied by a call to `r.param`.	constant
`r.param(ob)`	`void`	Copies to the given destination a sequence of 32-bit units that can be provided to the constructor of a second object of type `S`, and that would reproduce in that second object a state indistinguishable from the state of the first object.	$\mathcal{O}$(`r.size()`)

30.6.2.3 Uniform random bit generator requirements [rand.req.urng]

1 A *uniform random bit generator* `g` of type `G` is a function object returning unsigned integer values such that each value in the range of possible results has (ideally) equal probability of being returned. [*Note:* The degree to which `g`'s results approximate the ideal is often determined statistically. — *end note*]

```
template <class G>
concept UniformRandomBitGenerator =
  Invocable<G&> && UnsignedIntegral<invoke_result_t<G&>> &&
  requires {
    G::min(); requires Same<decltype(G::min()), invoke_result_t<G&>>;
    G::max(); requires Same<decltype(G::max()), invoke_result_t<G&>>;
  };
```

2 Let `g` be an object of type `G`. `G` models `UniformRandomBitGenerator` only if

(2.1) — Both `G::min()` and `G::max()` are constant expressions (8.6).

(2.2) — `G::min() < G::max()`.

(2.3) — `G::min() <= g()`.

(2.4) — `g() <= G::max()`.

(2.5) — `g()` has amortized constant complexity.

3 A class `G` meets the *uniform random bit generator* requirements if `G` models the `UniformRandomBitGenerator` concept, and additionally provides a nested *typedef-name* `result_type` that denotes the same type as `invoke_result_t<G&>`.