Recruiting test

Test assignment

Thank you for participating in our recruiting test. This will be a C++ programming test!

How to prepare for this test

Task Description

interval_map<K,V> is a data structure that efficiently associates intervals of keys of type K with values of type V. Your task is to implement the assign member function of this data structure, which is outlined below.

interval_map<K, V> is implemented on top of std::map. In case you are not entirely sure which functions std::map provides, what they do and which guarantees they provide, we provide an excerpt of the C++ standard here: More

Each key-value-pair (k,v) in the std::map means that the value v is associated with the interval from k (including) to the next key (excluding) in the std::map.

Example: the std::map (0,'A'), (3,'B'), (5,'A') represents the mapping

0 -> 'A'
1 -> 'A'
2 -> 'A'
3 -> 'B'

7 -> 'A'
... all the way to numeric_limits<int>::max()
The representation in the std::map must be canonical, that is, consecutive map entries must not have the same value: ..., (0,'A'), (3,'A'), ... is not allowed. Initially, the whole range of K is associated with a given initial value, passed to the constructor of the interval_map<K,V> data structure.

Key type K

- besides being copyable and assignable, is less-than comparable via operator<
- is bounded below, with the lowest value being std::numeric_limits<K>::lowest()
- does not implement any other operations, in particular no equality comparison or arithmetic operators

Value type V

- besides being copyable and assignable, is equality-comparable via operator==
- does not implement any other operations

You are given the following source code:

```
#include <map>
#include <limits>

template<typename K, typename V>
class interval_map {
        std::map<K,V> m_map;

public:
    // constructor associates whole range of K with val by inserting (K_min, v
    // into the map
    interval_map( V const& val) {
        m_map.insert(m_map.end(),std::make_pair(std::numeric_limits<K>::lowest
    }

    // Assign value val to interval [keyBegin, keyEnd).
    // Overwrite previous values in this interval.
    // Conforming to the C++ Standard Library conventions, the interval
    // includes keyBegin, but excludes keyEnd.
```

```
    void assign( K const& keyBegin, K const& keyEnd, V const& val ) {
```

```
Please insert your solution here
```

```
    }

    // look-up of the value associated with key
    V const& operator[]( K const& key ) const {
        return ( --m_map.upper_bound(key) )->second;
    }
};

// Many solutions we receive are incorrect. Consider using a randomized test
// to discover the cases that your implementation does not handle correctly.
// We recommend to implement a test function that tests the functionality of
// the interval_map, for example using a map of unsigned int intervals to char
```

You can download this source code here: Download

- **Type requirements are met:** You must adhere to the specification of the key and value type given above.
- **Correctness:** Your program should produce a working interval_map with the behavior described above. In particular, pay attention to the validity of iterators. It is illegal to dereference end iterators. Consider using a checking STL implementation such as the one shipped with Visual C++ or GCC.
- **Canonicity:** The representation in m_map must be canonical.
- **Running time:** Imagine your implementation is part of a library, so it should be big-O optimal. In addition:
  - Do not make big-O more operations on K and V than necessary, because you do not know how fast operations on K/V are; remember that constructions, destructions and assignments are operations as well.
  - Do not make more than two operations of amortized O(log N), in contrast to O(1), running time, where N is the number of elements in m_map. Any operation that needs to find a position in the map "from scratch", without being given a nearby position, is such an operation.
  - Otherwise favor simplicity over minor speed improvements.

You should not take longer than 9 hours, but you may of course be faster. Do not rush, we would not give you this assignment if it were trivial.

When you are done, please complete the form and click Compile. You can improve and compile solutions as often as you like.
Please submit your solution until 22:10 UTC.

Compile

Further instructions will be given once your code compiles correctly.