### **Task Description**

interval\_map<K,V> is a data structure that associates keys of type K with values of type V. It is designed to be used efficiently in situations where intervals of consecutive keys are associated with the same value. 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. For more information on std::map, you may refer to [cppreference.com](https://en.cppreference.com/w/cpp/container/map" \t "https://server.think-cell.com/portal/en/_blank).

Each key-value-pair (k,v) in interval\_map<K,V>::m\_map means that the value v is associated with all keys from k (including) to the next key (excluding) in m\_map. The member interval\_map<K,V>::m\_valBegin holds the value that is associated with all keys less than the first key in m\_map.

Example: Let M be an instance of interval\_map<int,char> where

* M.m\_valBegin=='A',
* M.m\_map=={ (1,'B'), (3,'A') },

Then M represents the mapping

* ...
* -2 -> 'A'
* -1 -> 'A'
* 0 -> 'A'
* 1 -> 'B'
* 2 -> 'B'
* 3 -> 'A'
* 4 -> 'A'
* 5 -> 'A'
* ...

The representation in the std::map must be canonical, that is, consecutive map entries must not contain the same value: ..., (3,'A'), (5,'A'), ... is not allowed. Likewise, the first entry in m\_map must not contain the same value as m\_valBegin. 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<, and
* 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==, and
* does not implement any other operations.