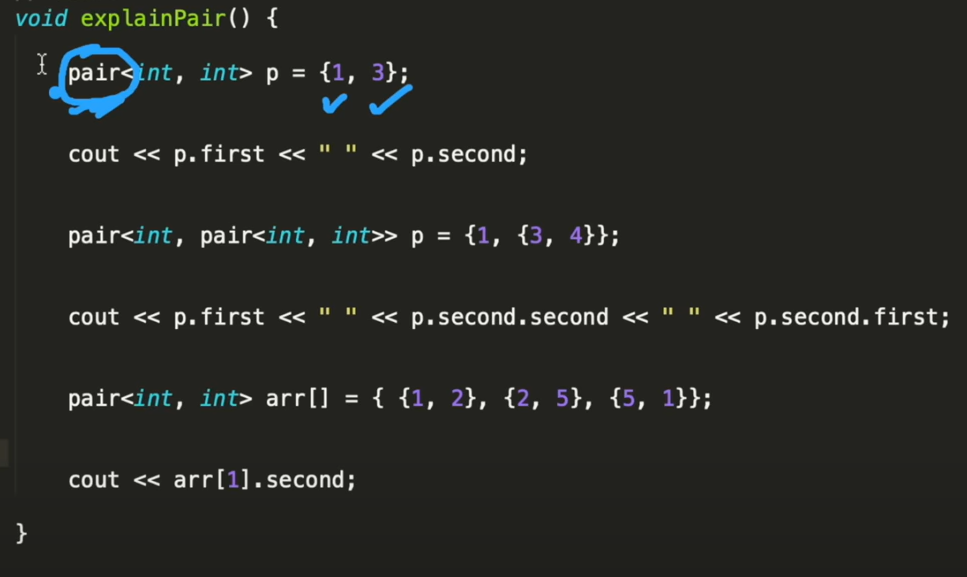
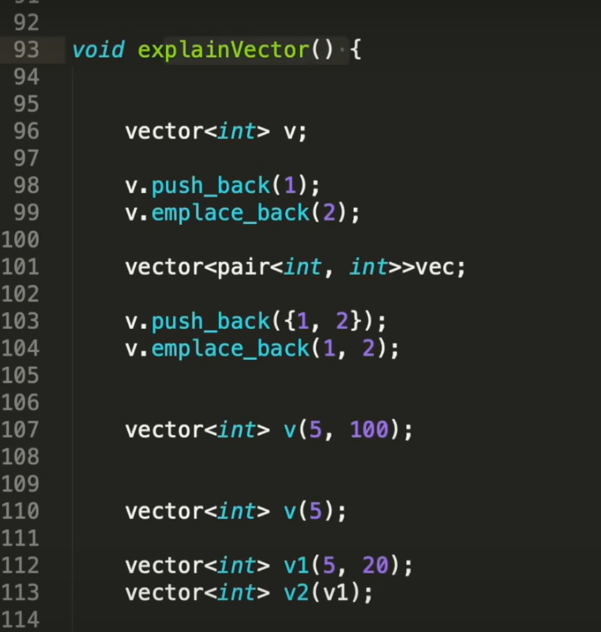
**C++ STL**

**Pairs:**



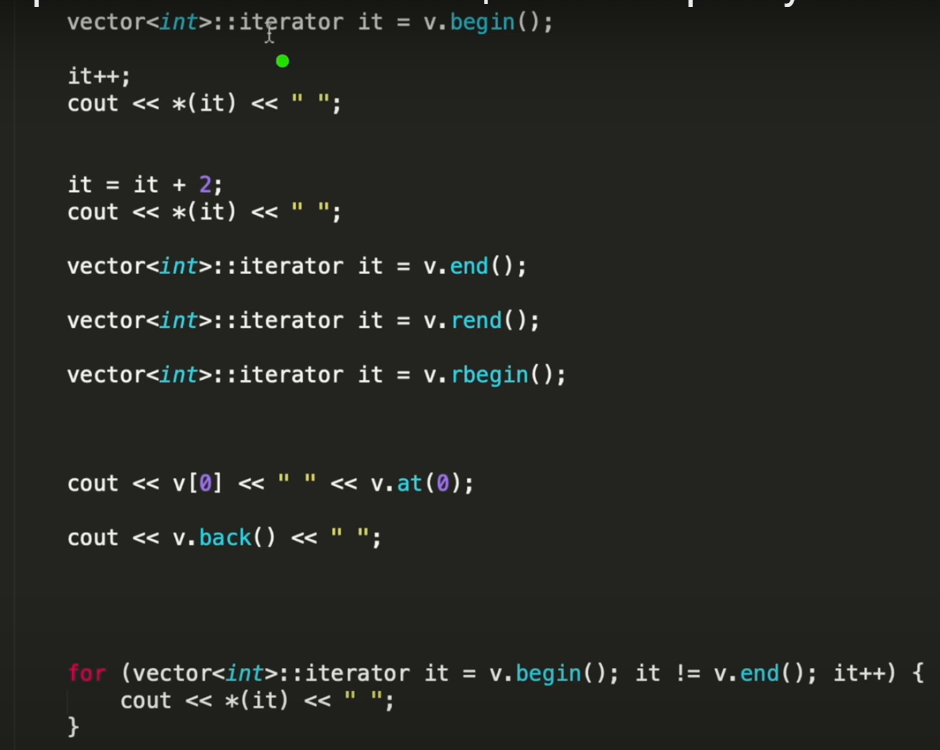
**Vector:**



v.end points to a address after the last element.

Iterator it here points to the address of the element

We need to dereference it using the \* , in order to get the value at that element



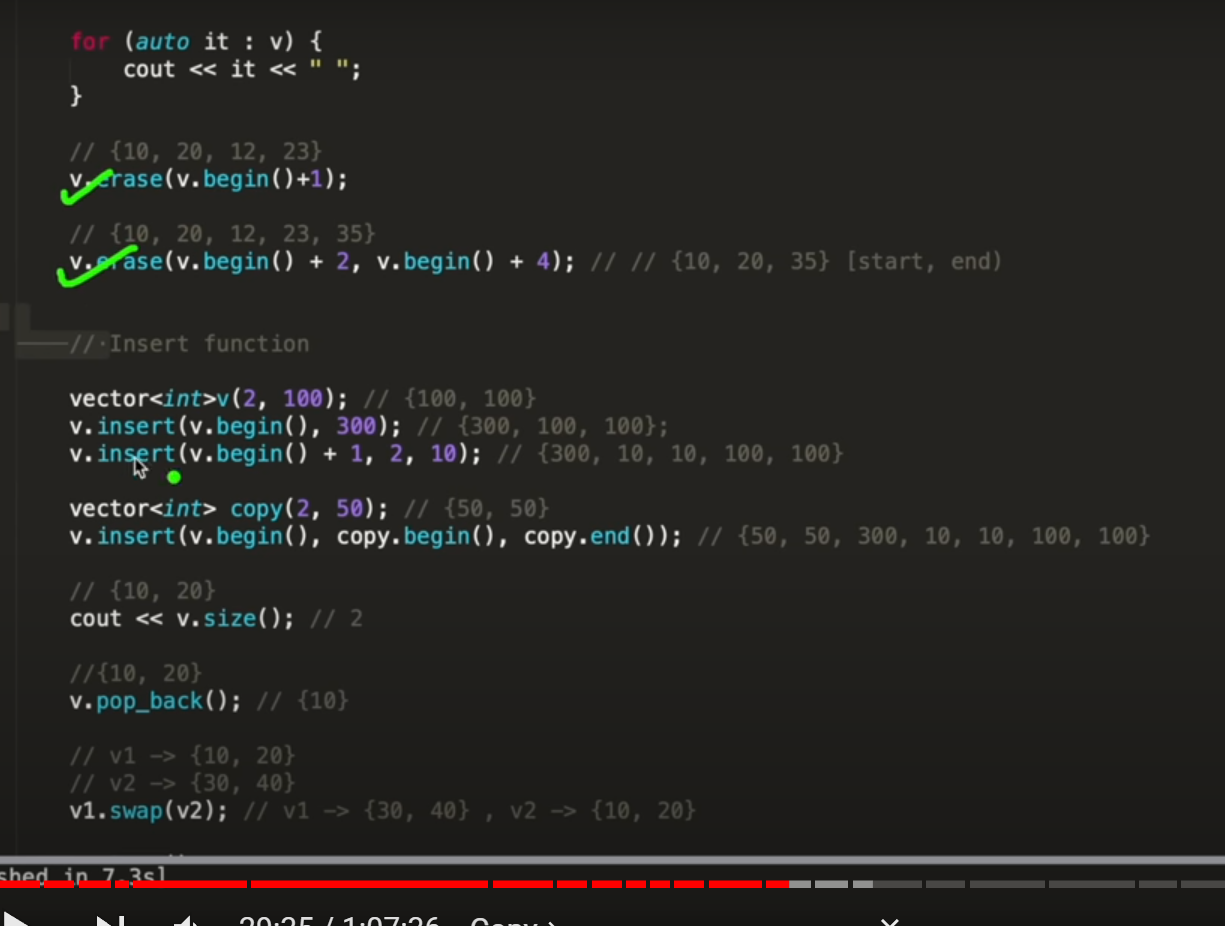
In the next page:

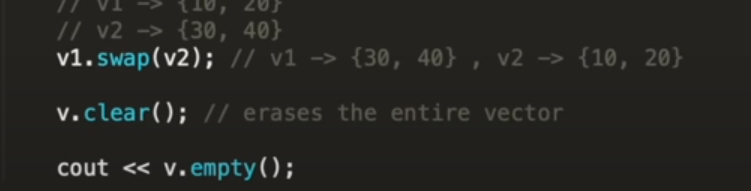
v.erase from start till where

suppose array is 10,20,30,40,50 and we want to delete 20 and 30

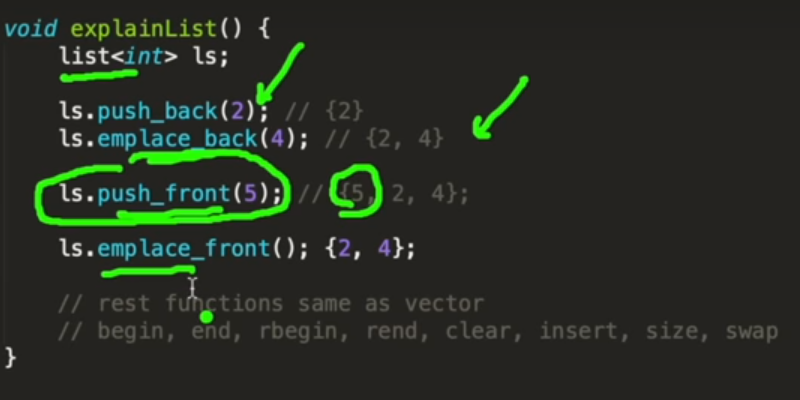
position of 20 is v.begin()+1 and we want to delete till 40(not 40 tho), 40 is v.begin()+3

v.erase(v.begin()+1,v.begin()+3)



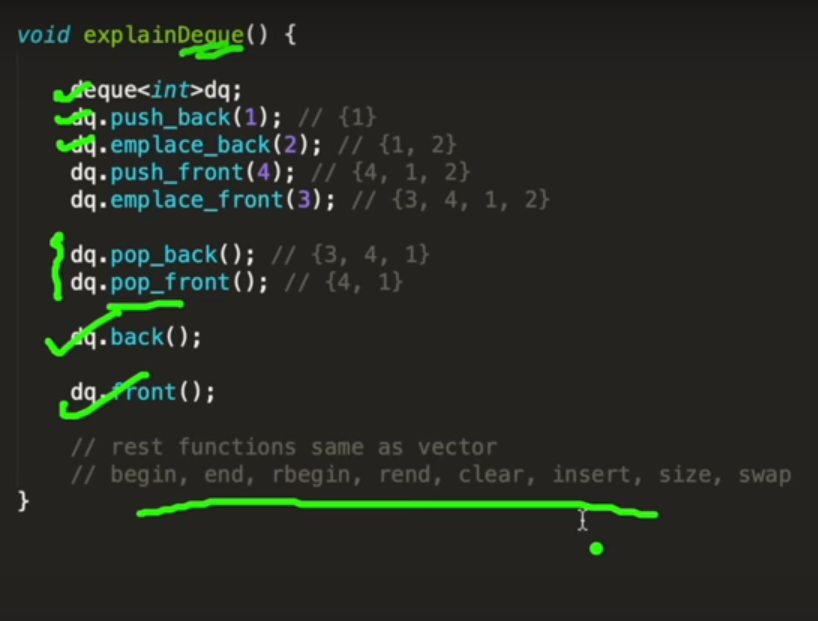


**List:**

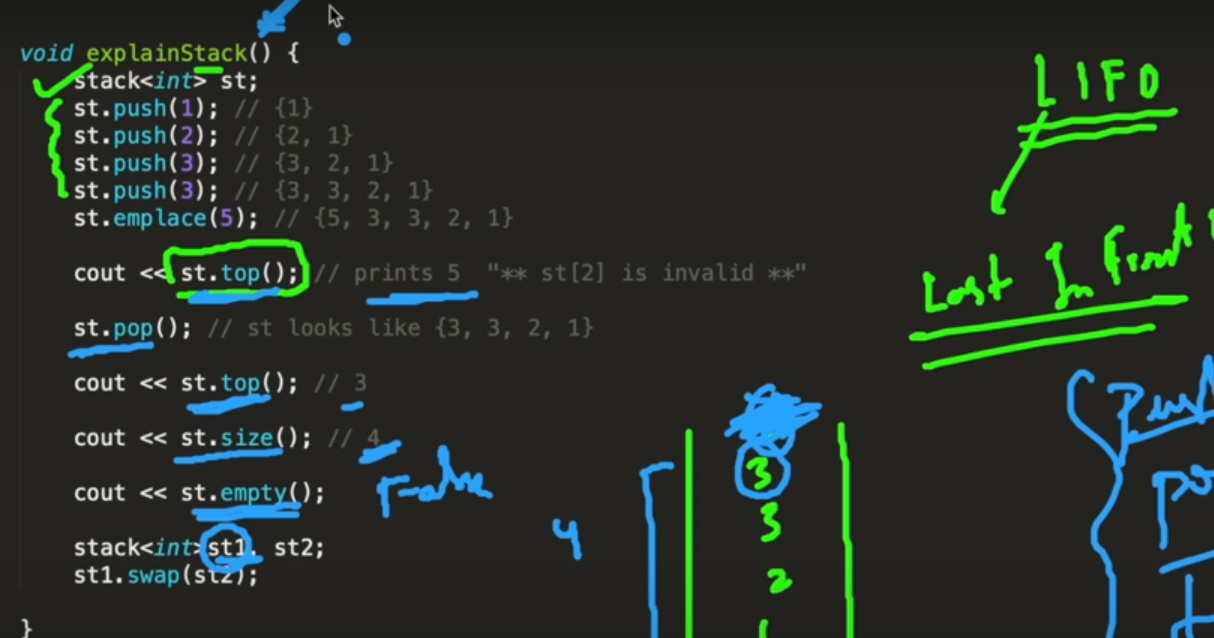


**Dequeue:** Same as vector, except front operations are allowed, in vector we have insert operation but that is very costly

In vector we use a singly ll, whereas in list we use doubly ll.

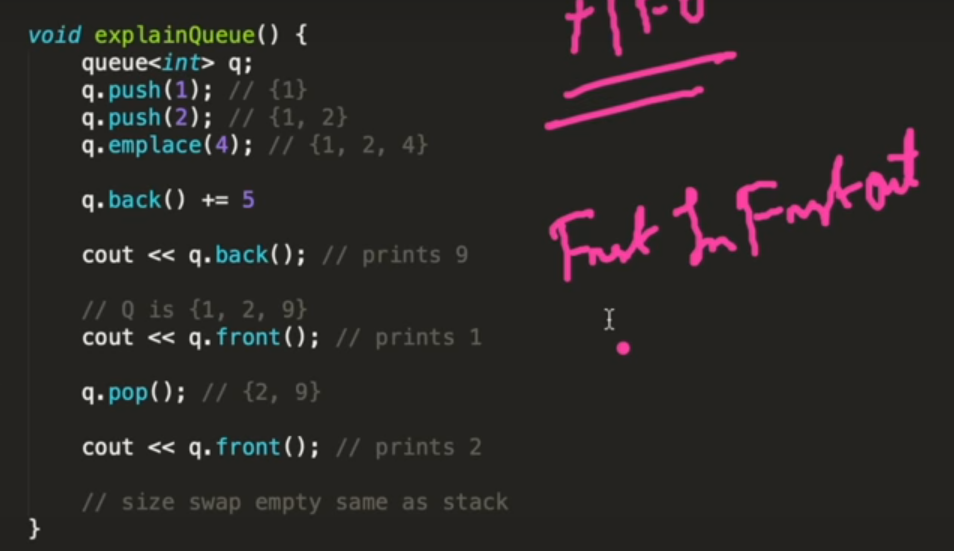


**Stack:**



TC of all operations in a stack is O(1).

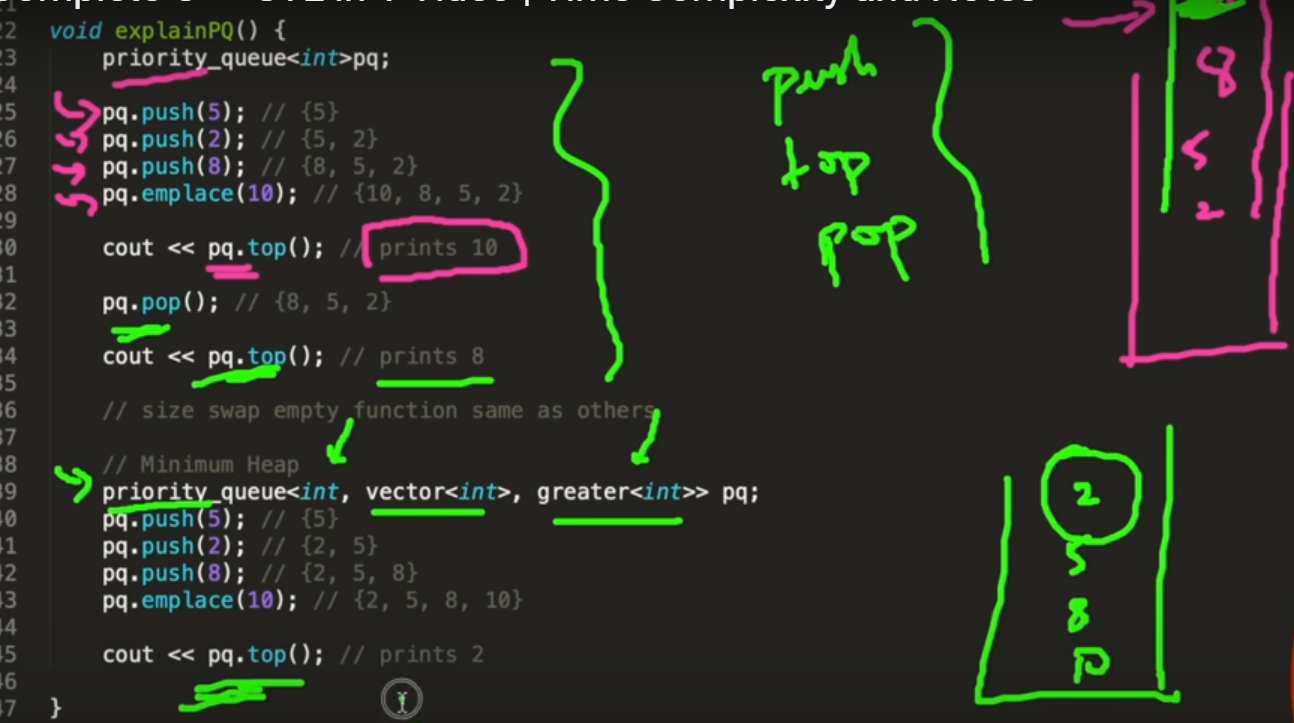
**Queue:**



All operations happen in constant time.

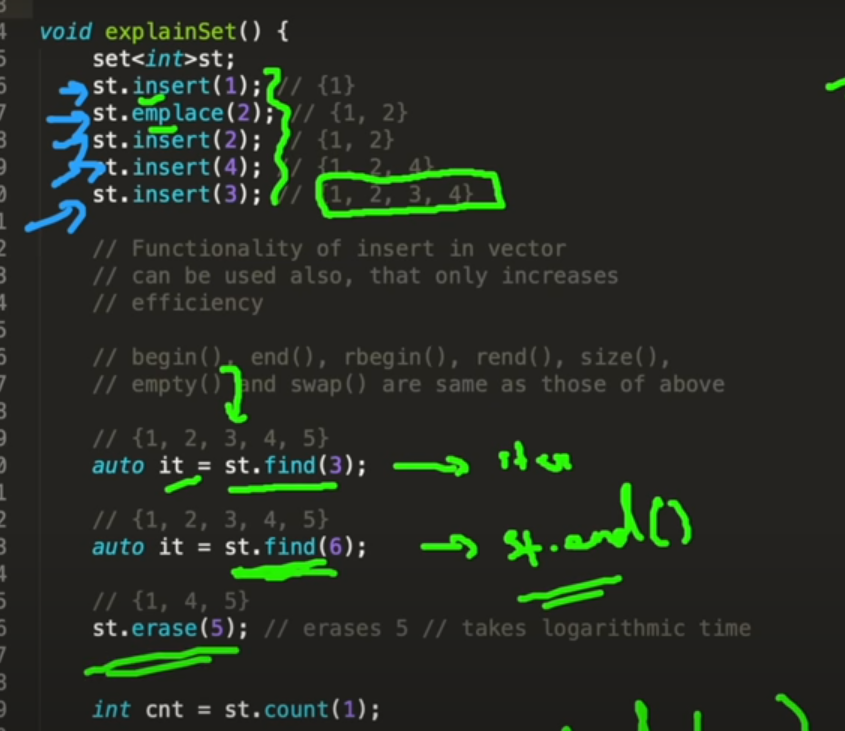
**Priority queue:** largest element will be on top and that will be popped first.

Its not a linear ds, a tree is maintained inside, which helps in storing the elements in order.



Push and pop: logn, top: O(1).

Set: Sorted and unique elements, a tree is maintained not linear

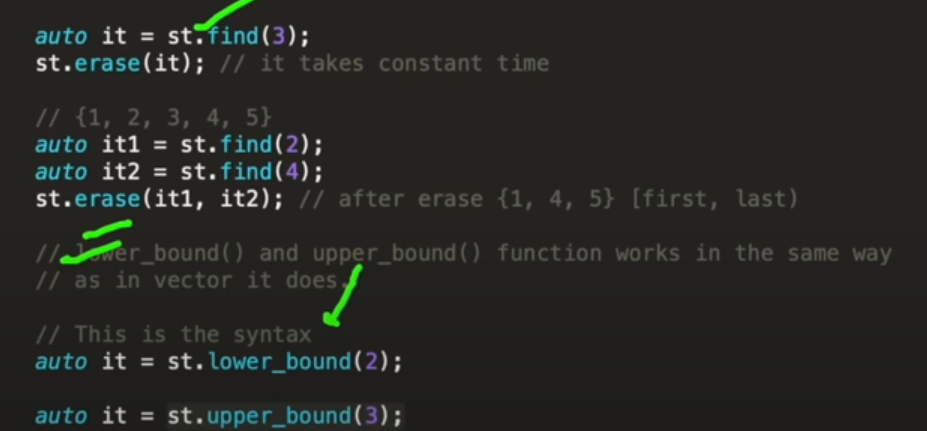


St.find returns an iterator to the element specified

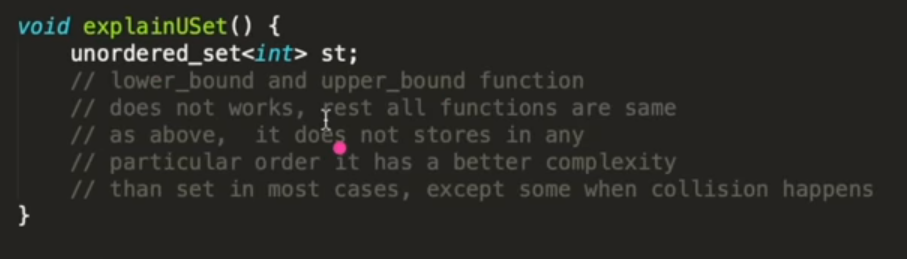
If the element is not present , then it returns an iterator which points to a address which is next to the last address.

In st.erase , we can specify the element to be deleted or its address

St.count will return 1 if the element is present, 0 otherwise, all operation logn



Unordered set: O(1) tc, worst case: very rare: O(1), only stores unique elements, random order



Map: key value pairs, kays and values can be any ds

Key should be unique cant have duplicates, but values can

Its stored in map in sorted order according to their keys

Sorting: The beg and end index, with ending not included

