**structure** node_t {value: data type, next: **pointer to** node_t}
**structure** queue_t {Head: **pointer to** node_t, Tail: **pointer to** node_t, H_lock: lock type, T_lock: lock type}

```
initialize(Q: pointer to queue_t)
        node = new_node()
        node–>next.ptr = NULL
        Q–>Head = Q–>Tail = node
        Q–>H_lock = Q–>T_lock = FREE

enqueue(Q: pointer to queue_t, value: data type)
        node = new_node()
        node–>value = value
        node–>next.ptr = NULL
        lock(&Q–>T_lock)
            Q–>Tail–>next = node
            Q–>Tail = node
        unlock(&Q–>T_lock)
```

```
from synch import Lock, acquire, release
from alloc import malloc, free

def Queue():
  let node = malloc({.next: None }):
    result = { .Head: node, .Tail: node,
               .H_lock: Lock(), .T_lock: Lock() }

def enqueue(Q, value):
  let node = malloc({ .value: value, .next: None }):
    acquire(?Q–>T_lock)
    Q–>Tail –>next = node
    Q–>Tail = node
    release(?Q–>T_lock)
```