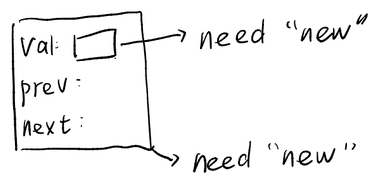
Solutions: Part 1

(1) Head guarder

(2) Each Node contain a value which needed to be claimed by "new" first. (it is a pointer point to an memory)

Node.




(3) pos→val is a pointer, use *(pos→val)

(4) (5) remember to delete

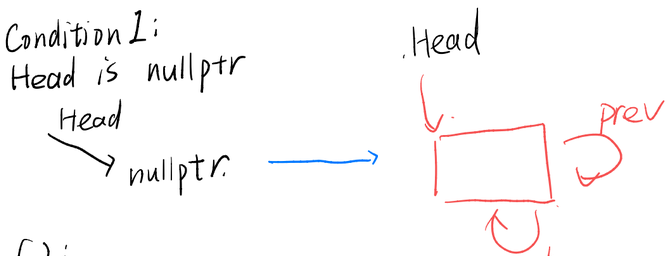
Part 2.

(2-a)

(2-b)

```
template <class T>
void DCLL<T>::insert(T *val) {
   Node *newNode = new Node;
   newNode->val = val;
   if (this->isEmpty()) {
      // (2-a) To Do ...
   } else {
      // (2-b) To Do ...
   }
   this->head = newNode;
}
```

Condition 1:
Head is nullptr




NewNode→next = NewNode

NewNode→prev = NewNode

C2:


NewNode→next = Head.

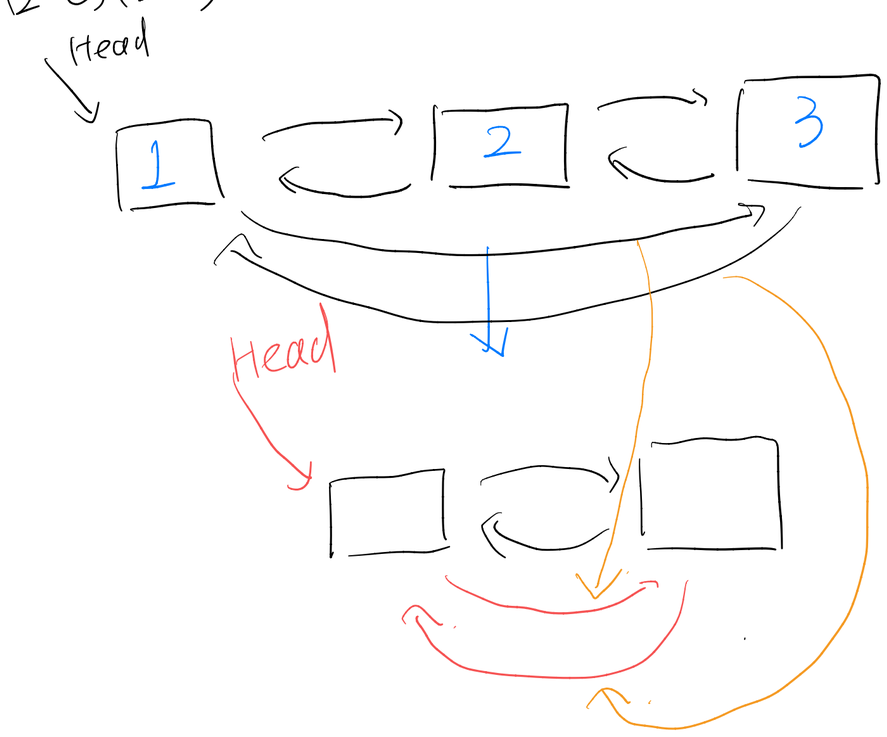
NewNode→prev = Head→prev. (tail!)

Head→prev→next = NewNode.

Head→prev = NewNode.

Head = NewNode.

(2-c) (2-d)




Head→prev→next = Head→next.

Head→next→prev = Head→prev

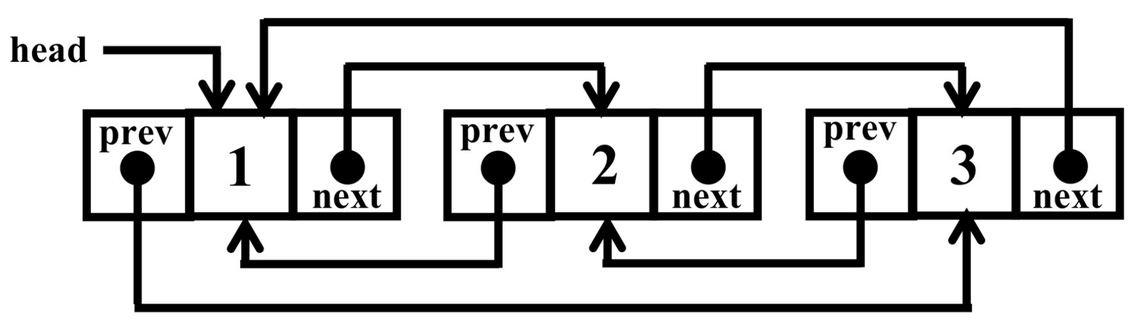
Head = Head→Next. (if No Node?)

Head need delete.

Part 3 (a)

Overload "<<"

(b)




Next of 1 → 2. Next of 2 → 3 Next of 3 → 1

Val 1 val 2

prev of 1 → 3. prev of 2 → 1 prev of 3 → 2.

Part 4: use circular-Link-List. implemented above

Part 5: read code carefully