```
public class LRUCache {
    Map<Integer, DoubleLinkedListNode> map = new HashMap<Integer,
DoubleLinkedListNode>();
    private DoubleLinkedListNode head;
    private DoubleLinkedListNode end;
    private int capacity;
    private int len;

    public LRUCache(int capacity) {
        this.capacity = capacity;
        len = 0;
    }

    public int get(int key) {
        if(map.containsKey(key))
        {
            DoubleLinkedListNode latest = map.get(key);
            removeNode(latest);
            setHead(latest);
            return map.get(key).val;
        }
        else
            return -1;
    }

    public void removeNode(DoubleLinkedListNode node)
    {
        DoubleLinkedListNode cur = node;
        DoubleLinkedListNode pre = cur.pre;
        DoubleLinkedListNode post = cur.post;

        if(pre != null)
        {
            pre.post = post;
        }
        else
        {
            head = post;
        }

        if(post != null)
        {
            post.pre = pre;
```

```
    }
    else
    {
        end = pre;
    }
}

public void setHead(DoubleLinkedListNode node)
{
    node.post = head;
    node.pre = null;
    if(head != null)
    {
        head.pre = node;
    }
    head = node;
    if(end == null)
    {
        end = node;
    }
}
public void set(int key, int value) {
    if(map.containsKey(key))
    {
        DoubleLinkedListNode oldNode = map.get(key);
        oldNode.val = value;
        removeNode(oldNode);
        setHead(oldNode);
    }
    else
    {
        DoubleLinkedListNode newNode = new DoubleLinkedListNode(value, key);
        if(len < capacity)
        {
            setHead(newNode);
            map.put(key, newNode);
            len++;
        }
        else
        {
            map.remove(end.key);
            end = end.pre;
            if(end != null)
```

```
            {
                end.post = null;
            }
            setHead(newNode);
            map.put(key, newNode);
        }

    }
  }
}

class DoubleLinkedListNode
{
   public DoubleLinkedListNode pre;
   public DoubleLinkedListNode post;
   public int val;
   public int key;

   public DoubleLinkedListNode(int val, int key)
   {
      this.val = val;
      this.key = key;
   }
}
```