Implementation of LRU using

Dictionary (key, Value-pointers)

Doubly linked list (value)

I have used doubly linked list over deque, list or singly linked list because of its time complexity 0(1) when removing item between the list.

Please execute attached files in following order

1. dllisy.py
2. LRUCache.py
3. Test\_LRUCache.py

Algorithm (Explaining output from TestLRU\_Cache) :-

Set length of cache: 3

Push new item (1,”One”) in cache: **O(1)**

Create new head node in doubly link list

Insert new key, value pair in dictionary

Push new item (2,”Two”) in cache: **O(1)**

Create new node

Set Head=new node

Insert new key, value pair in dictionary

Push new item (3,”Three”) in cache: **O(1)**

Create new node

Set Head=new node

Insert new key, value pair in dictionary

Push existing item with new value (2,”Five”) in cache: **O(1)**

Get node pointer from Dictionary

Remove that pointer from doubly link list

Create new node

Set Head=new node

Insert new key, value pair in dictionary

Push new item with value (4,”Four”) in cache: **O(1)**

Dictionary.Length = cache.lenth

Pop last node from doubly link list

Pop node(key,value) from dictionary

Create new node

Set Head=new node

Insert new key, value pair in dictionary

Get itemwith value (2,”Five”) in Cache: **O(1)**

Get node pointer from Dictionary

Remove that pointer from doubly link list

Set Head=returned node