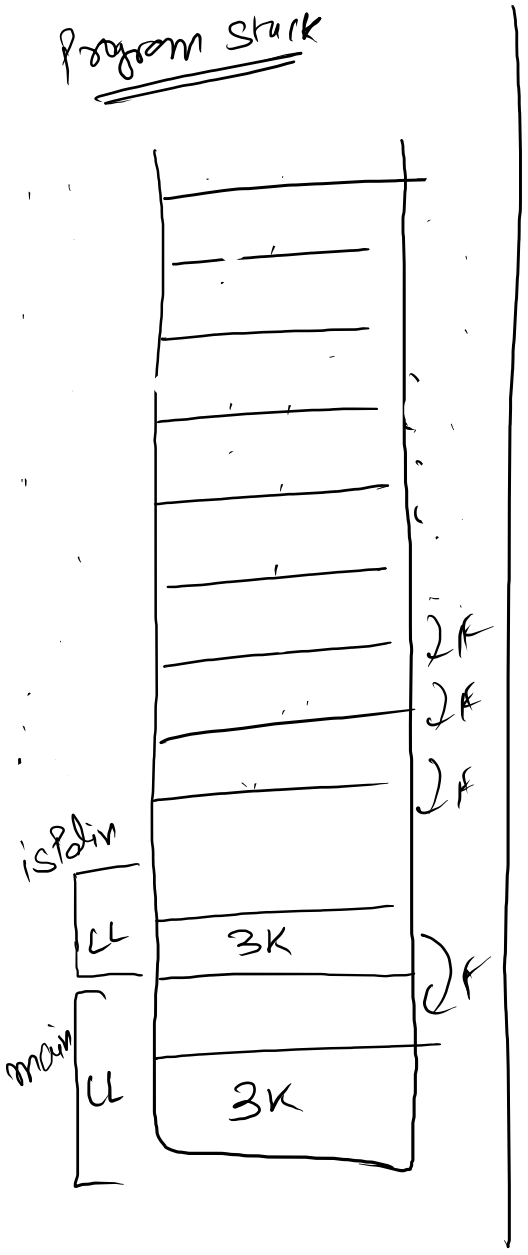


head = 4K

tail = 11K

size = 8

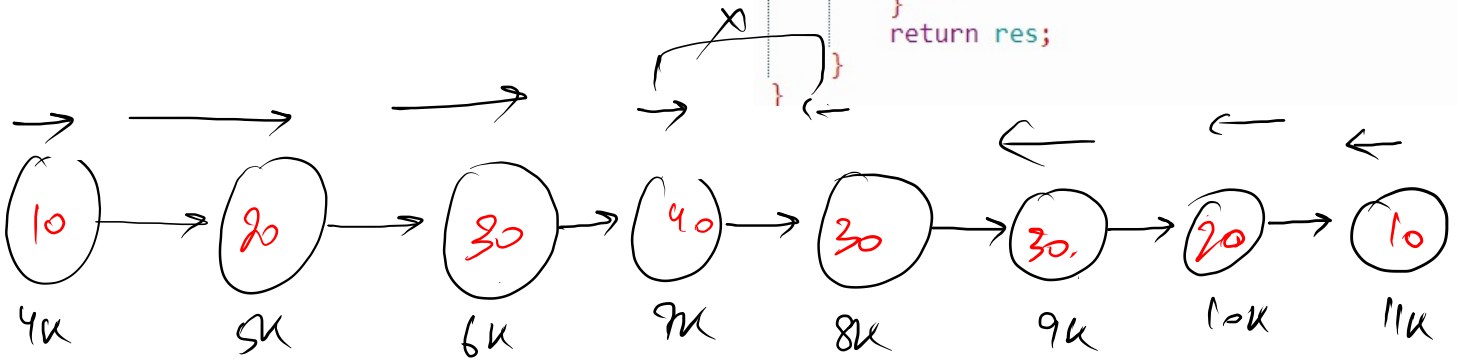
3K

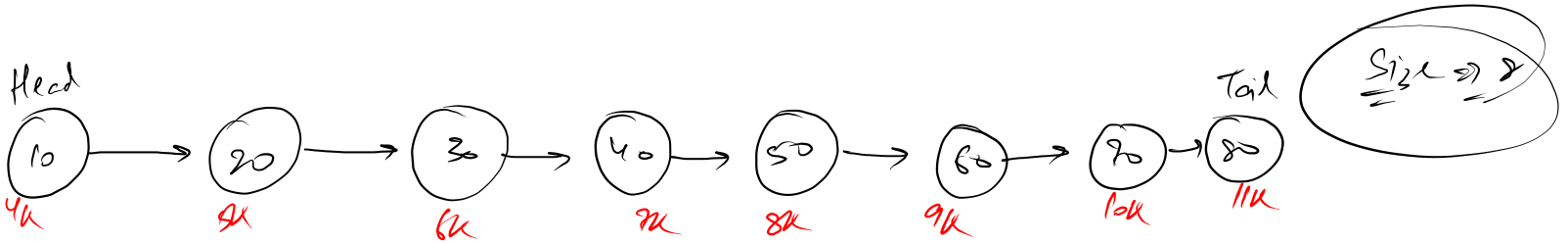
left = 7K

```
public boolean IsPalindrome() {
  left = this.head;
  return IsPalindromeHelper(this.head,0);
}
static Node left;
public boolean IsPalindromeHelper(Node node,int idx){
    if(node == null){
        return true;
    }

    boolean res = IsPalindromeHelper(node.next,idx+1);

    if(res == false){
        return false;
    }else{
        if(idx >= this.size / 2){
            if(left.data == node.data){
                left = left.next;
                return true;
            }else{
                return false;
            }
        }
        return res;
    }
}
```





H.W.

fwd ← left.next

left.next = right;

right.next = fwd;

left = fwd;

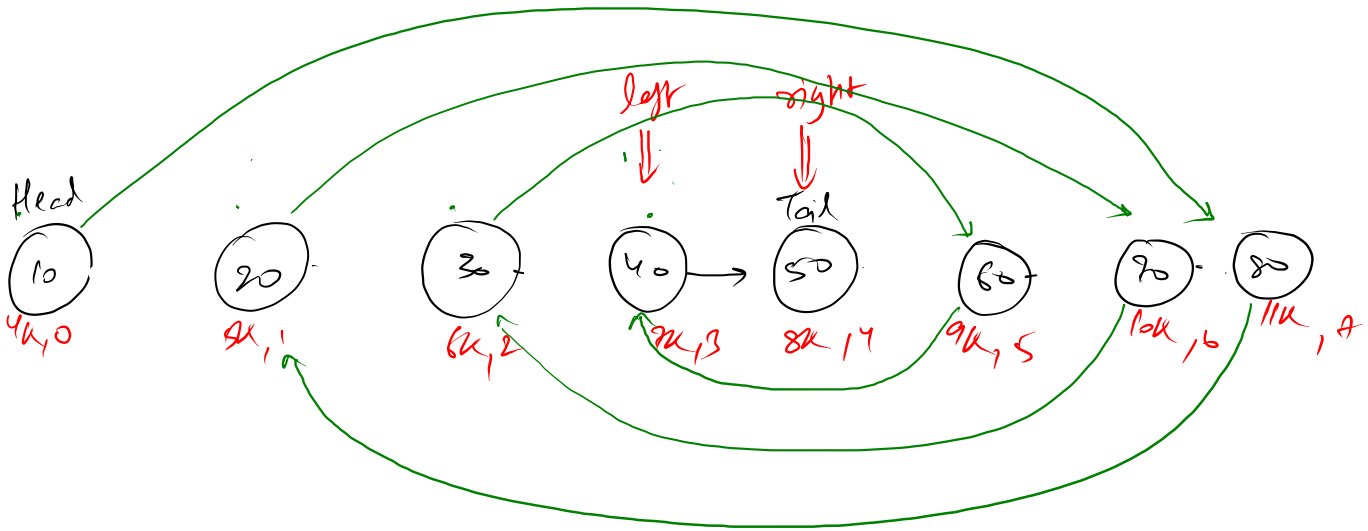
7, 6, 5

if(idx > size/2)

4

if(idx == size/2) { right.next = null; Tail = right }

Size: 8

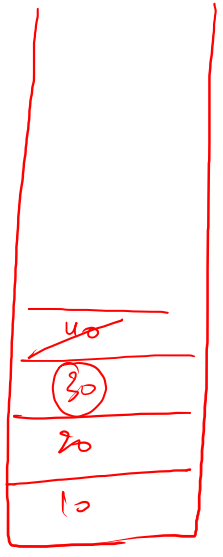


Stack → ① Linear

② LIFO

③ Operations
- push
- pop
- peek
- size

- ✓ push 10
- ✓ push 20
- ✓ push 30
- ✓ push 40
- pop → 40
- peek →
  - pop
  - peek
  - pop
  - peek



h : 5k
r : m
s : 4

Java Implementation → DLL

Class Implementation → SLL

add Last	add First ✓
remove Last	remove first ✓
get Last	get First ✓
①	②

SLL → ②

DLL → ① / ②

SLL
- get First → O(1)
- get Last → O(1)
- get At → O(n)
- remove First → O(1)
- remove Last → O(n) → SLL, O(1) → DLL
- remove At → O(n)
- add First → O(1)
- add Last → O(1)
- add At → O(n)
- Size → O(1)

Queue →

add First	add Last
remove Last	remove first
get Last	get First
①	②

SLL → ②

DLL → ①

```java
public static class LLToQueueAdapter {
  LinkedList<Integer> list;

  public LLToQueueAdapter() {
    list = new LinkedList<>();
  }

  int size() {
    return list.size();
  }

  void add(int val) {
    list.addLast(val);
  }

  int remove() {
    if(list.size() == 0){
        System.out.println("Queue underflow");
        return -1;
    }
    return list.removeFirst();
  }

  int peek() {
    if(list.size() == 0){
        System.out.println("Queue underflow");
        return -1;
    }
    return list.getFirst();
  }
}
```

```java
public static class LLToStackAdapter {
  LinkedList<Integer> list;

  public LLToStackAdapter() {
    list = new LinkedList<>();
  }

  int size() {
    return list.size();
  }

  void push(int val) {
    list.addFirst(val);
  }

  int pop() {
    if(list.size() == 0){
        System.out.println("Stack underflow");
        return -1;
    }
    return list.removeFirst();
  }

  int top() {
    if(list.size() == 0){
        System.out.println("Stack underflow");
        return -1;
    }
    return list.getFirst();
  }
}
```