class Deque:  
 # 0(1)  
 def \_\_init\_\_(self):  
 self.deque\_list = []  
  
 # O(1)  
 def add\_front(self,front\_item):  
 self.deque\_list.insert(0, front\_item)  
 # O(1)  
 def add\_rear(self,rear\_item):  
 self.deque\_list.append(rear\_item)  
 # O(1)  
 def remove\_front(self):  
 first\_index = self.deque\_list.pop(0)  
 return first\_index  
 # O(n)  
 def remove\_rear(self):  
 last\_index = len(self.deque\_list)  
 self.deque\_list.pop(last\_index-1)  
 return last\_index  
 # O(n)  
 def size(self):  
 return len(self.deque\_list)  
  
 # O(n)  
 def \_\_str\_\_(self):  
 print\_string = "( "  
 if self is None:  
 return "[]"  
 else:  
 for i in range(self.size()):  
 worlds\_most\_annoying\_int = self.deque\_list[i]  
 print\_string += str(worlds\_most\_annoying\_int) + " "  
 print\_string += ")"  
 return print\_string  
 # O(1)  
 def is\_empty(self):  
 return self.size() == 0  
 # O(n)  
 def test\_str(self):  
 return self.deque\_list[len(self.deque\_list)-1]  
  
  
  
new\_list = Deque()  
for x in range(10):  
 new\_list.add\_front(x)  
  
print(new\_list)  
new\_list.add\_front(12)  
print(new\_list)  
new\_list.add\_rear(4)  
print(new\_list)  
new\_list.remove\_front()  
print(new\_list)  
new\_list.remove\_rear()  
print(new\_list)  
  
print(new\_list.size())  
print(new\_list.is\_empty())