Lab 3 Report Template

1. In coffee can, there are n black beans and m white beans, each time two beans will be taken out: if they are with same color, throw away both, then put in one black bean; if they are with different color, throw away the black bean, put back the white bean.

- 自己设计采用 list 结构表示每个 bean，存储其颜色(black/ white)，实现下表中 list 的接口
- 每次取法：产生两个随机数，取出对应 bean，删除相应节点
- 每次放回：加入到最后
- 取不同的(n,m)且每组值运行 10 次，输出实验过程和结果
  n、m 分别取 50-100（51*51 种组合）
- 采用 int 表示节点数据类型（不要直接采用网上的 template)

```
with open("./game_output.txt", "w") as file:

    for n in range(50, 101):
        for m in range(50, 101):
            data = ["Black"]*n+["White"]*m
            random.shuffle(data)
            file.write("Start:\t\t\t\tBalck:"+str(n)+"\tWhite:"+str(m)+"\n")
            print("Start:\t\t\t\tBalck:"+str(n)+"\tWhite:"+str(m)+"\n")
            coffee_can = Pylist()
            for bean in data:
                coffee_can.insert_as_first(bean)
            #print("change")
            turn = 1
            while turn < 11:
                ran_a = random.randint(0, coffee_can.size()-1)
                ran_b = random.randint(0, coffee_can.size()-1)
                if ran_b == ran_a:
                    ran_b = random.randint(0, coffee_can.size()-1)
                bean_a = coffee_can.remove(max(ran_a, ran_b))
                bean_b = coffee_can.remove(min(ran_a, ran_b))
                file.write("Turn"+str(turn)+":\t"+bean_a+","+bean_b+"\t")
                if bean_a == bean_b:
                    coffee_can.insert_as_last("Black")
                if bean_a != bean_b:
                    coffee_can.insert_as_last("White")
                balck = count(coffee_can, "Black")
                white = count(coffee_can, "White")
                file.write("Balck:"+str(balck)+"\tWhite:"+str(white)+"\n")
                turn += 1
            file.write("\n")
```

The result is stored in the file game_output.txt

For more information about the rewritten list class, please check the file:

coffee_can_pylist.py for source code.

2. 构造一组输入数据，存入 list 结构，调用 list 接口，依次调用测试下列接口

列表：ADT接口

操作接口	功能	适用对象
size()	报告列表当前的规模（节点总数）	列表
first(), last()	返回首、末节点的位置	列表
insertAsFirst(e), insertAsLast(e)	将e当作首、末节点插入	列表
insertA(p, e), insertB(p, e)	将e当作节点p的直接后继、前驱插入	列表
remove(p)	删除位置p处的节点，返回其引用	列表
disordered()	判断所有节点是否已按非降序排列	列表
sort()	调整各节点的位置，使之按非降序排列	列表
find(e)	查找目标元素e，失败时返回NULL	列表

the test program in the left and the output in the right:

The source code: pylist.py

```
exsample = Pylist(0,1,2,3,4,5,6,7,8,10,9)

print(str(exsample))
print(exsample.size())
print(exsample.first())
print(exsample.last())

exsample.insert_as_first(100)
print(str(exsample))

exsample.insert_as_last(200)
print(str(exsample))

exsample.insertA(4,300)
print(str(exsample))

exsample.insertB(7,400)
print(str(exsample))

print(exsample.remove(7))

print(str(exsample))

print(exsample.disordered())
exsample.sort()
print(str(exsample))
print(exsample.disordered())

print(exsample.find(100))
print(exsample.find(700))
```

```
0,1,2,3,4,5,6,7,8,10,9
11
0
9
100,0,1,2,3,4,5,6,7,8,10,9
100,0,1,2,3,4,5,6,7,8,10,9,200
100,0,1,2,300,3,4,5,6,7,8,10,9,200
100,0,1,2,300,3,4,5,400,6,7,8,10,9,200
5
100,0,1,2,300,3,4,400,6,7,8,10,9,200
False
0,1,2,3,4,6,7,8,9,10,100,200,300,400
True
10
False
```

类别	排序方法	时间复杂度			空间复杂度
		平均情况	最好情况	最坏情况	辅助存储
插入排序	直接插入	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$
	shell排序	$O(n^{1.3})$	$O(n)$	$O(n^2)$	$O(1)$
选择排序	直接选择	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$
	堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(1)$
交换排序	冒泡排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$
	快速排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n^2)$	$O(n\log_2 n)$
归并排序		$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(1)$

学会分析算法的时间复杂度。代码实现 7 种排序算法（可参考网上资料），并给出不同规模（1000,10000,10000）下数字排序的运行时间

Source code: 7kind_sort_test.py

sort algorithms	Time(s) for different sizes		
	100,0	100,00	100,00
Insert sort 直接插入排序	0.021037	2.25501	2.44149
Shell sort shell 排序	0.070585	9.08224	9.67151
Select sort 简单选择排序	0.043516	4.39137	4.60550
Heap sort 堆排序	0.005799	0.11467	0.10802
Bubble sort 冒泡排序	0.043715	4.40686	4.56962
Quick sort 快速排序	0.059198	5.48732	5.49152
Merge sort 归并排序	3.099441 e-06	2.86102 e-06	3.09944 e-06