比特就业课Java方向笔试强训48天day21_11月9日（105Java班）-王世国-测评结果

考生信息 存在作弊行为

王世国

投递编号：91 | 学校：武汉轻工大学 | 邮箱：1477649017@qq.com | 职位：比特就业课105期Java2班 |

参考区域: 湖北武汉（113.57.53.192） |

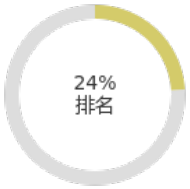
做题用时：03:33:29(2022-11-08 21:55:55开始答题，2022-11-09 11:17:55交卷) | 作答设备： PC |

已同意诚信声明和隐私协议

83.75 分 / 100分

在本次考试中，考生总成绩为83.75分/100分，评级为B（排名前24%），编程能力良好（1题通过，1题部分通过，分数排名前38%），编程思路基本一致，编程规范性高。该考生在本次考试中存在作弊行为，视频监控图片数目过少，有遮挡或关闭摄像头的嫌疑。

考生成绩



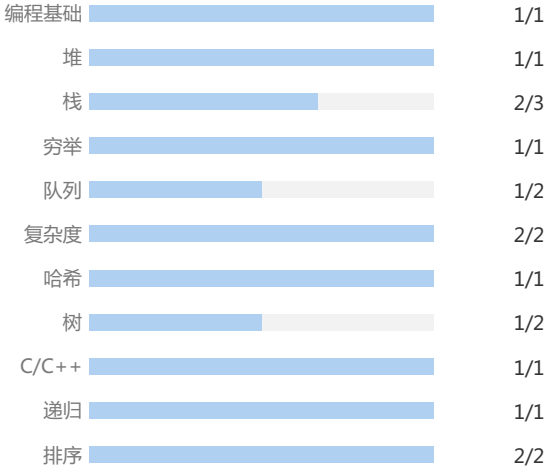
题型	得分	正确题数	排名	用时	是否阅卷
单选	40.0	8	16	00:24:46	已阅
编程	43.75	1	35	02:37:50	已阅

作弊风险

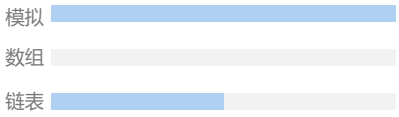
高风险 未开启摄像头

考生可能关闭摄像头，导致视频截图较少

知识点技能图谱



知识点	得分	正确题数
编程基础	5.0	1
堆	5.0	1
栈	10.0	2
穷举	25.0	1
队列	5.0	1
复杂度	10.0	2



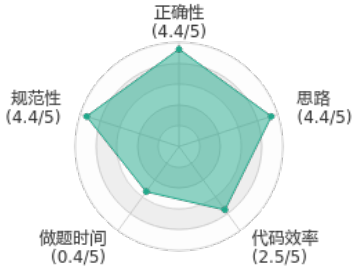
知识点	得分	正确题数
哈希	5.0	1
树	5.0	1
C/C++	5.0	1
递归	5.0	1
排序	30.0	2
模拟	25.0	1
数组	18.75	0
链表	5.0	1

历史笔试记录

序号	试卷名称	排名	总得分	得分详情	作弊嫌疑	安排笔试时间	交卷时间
1	比特就业课105期+2022寒假班C1考试	20.0%	48.8/60	单选:30.0分 编程:18.75分	否	2022-03-29 11:16:18	2022-03-31 18:51:27
2	比特就业课105期+2022寒假班C2考试	66.0%	24.0/60	单选:24.0分 编程:0.0分	否	2022-04-11 14:12:23	2022-04-11 20:12:16
3	比特就业课 105期JavaSE考试	11.0%	56.0/60	单选:26.0分 编程:30.0分	否	2022-07-12 16:00:16	2022-07-13 15:48:42
4	比特就业课 105期java方向 数据结构考试	47.0%	50.0/60	单选:20.0分 编程:30.0分	否	2022-07-23 12:49:22	2022-07-25 09:56:28
5	比特就业课105期Java方向笔试强训48天day01_10月10日	36.0%	80.0/100	单选:40.0分 编程:40.0分	是，摄像头监控异常	2022-10-09 17:29:16	2022-10-09 21:57:25
6	比特就业课105期Java方向笔试强训48天day02_10月11日	13.0%	90.0/100	单选:40.0分 编程:50.0分	是，代码抄袭	2022-10-10 10:43:48	2022-10-10 21:13:15
7	比特就业课105期Java方向笔试强训48天day03_10月12日	1.0%	95.0/100	单选:45.0分 编程:50.0分	是，摄像头监控异常	2022-10-11 10:40:53	2022-10-12 10:03:09
8	比特就业课105期Java方向笔试强训48天day04_10月13日	2.0%	95.0/100	单选:30.0分 不定项选择:15.0分 编程:50.0分	是，摄像头监控异常	2022-10-12 10:31:10	2022-10-12 21:14:05
9	比特就业课105期Java方向笔试强训48天day05_10月14日	12.0%	85.0/100	单选:35.0分 不定项选择:0.0分 编程:50.0分	是，摄像头监控异常|代码抄袭	2022-10-13 11:41:43	2022-10-14 11:33:24

序号	试卷名称	排名	总得分	得分详情	作弊嫌疑	安排笔试时间	交卷时间
10	比特就业课105期Java方向笔试强训48天day06_10月15日	13.0%	86.7/100	单选:25.0分 不定项选择:11.67分 编程:50.0分	是，摄像头监控异常	2022-10-14 10:59:38	2022-10-14 21:18:11
11	比特就业课105期Java方向笔试强训48天day07_10月17日	1.0%	100.0/100	单选:50.0分 编程:50.0分	是，摄像头监控异常	2022-10-16 16:46:53	2022-10-16 20:38:39
12	比特就业课105期Java方向笔试强训48天day08_10月18日	33.0%	87.5/100	单选:40.0分 编程:47.5分	是，摄像头监控异常	2022-10-17 16:18:42	2022-10-17 20:16:45
13	比特就业课105期Java方向笔试强训48天day09_10月19日	3.0%	95.0/100	单选:45.0分 编程:50.0分	是，摄像头监控异常	2022-10-18 17:07:17	2022-10-19 11:02:55
14	比特就业课105期Java方向笔试强训48天day10_10月20日	58.0%	65.0/100	单选:40.0分 编程:25.0分	是，摄像头监控异常	2022-10-19 15:29:54	2022-10-19 21:44:34
15	比特就业课105期Java方向笔试强训48天day11_10月21日	2.0%	95.0/100	单选:45.0分 编程:50.0分	是，摄像头监控异常	2022-10-20 16:46:09	2022-10-20 23:36:15
16	比特就业课105期Java方向笔试强训48天day12_10月22日	15.000001%	90.0/100	单选:40.0分 编程:50.0分	是，摄像头监控异常	2022-10-21 16:44:18	2022-10-21 22:49:24
17	比特就业课105期Java方向笔试强训48天day13_10月24日	31.0%	65.0/100	单选:40.0分 编程:25.0分	是，摄像头监控异常|代码抄袭	2022-10-22 16:07:25	2022-10-23 23:15:22
18	比特就业课105期Java方向笔试强训48天day14_10月25日	58.999996%	55.0/100	单选:30.0分 编程:25.0分	是，摄像头监控异常	2022-10-24 14:08:32	2022-10-24 23:42:01
19	比特就业课105期Java方向笔试强训48天day15_10月26日	42.0%	55.0/100	单选:30.0分 编程:25.0分	是，摄像头监控异常	2022-10-24 14:11:25	2022-10-25 22:58:05
20	比特就业课105期Java方向笔试强训48天day16_10月27日	13.0%	89.4/100	单选:45.0分 编程:44.44分	是，摄像头监控异常	2022-10-24 14:25:44	2022-10-26 21:33:12
21	比特就业课105期Java方向笔试强训48天day17_10月28日	24.0%	83.3/100	单选:40.0分 编程:43.33分	是，摄像头监控异常	2022-10-24 14:29:13	2022-10-28 22:18:13
22	比特就业课105期Java方向笔试强训48天day18_10月29日	12.0%	84.1/100	单选:40.0分 编程:44.12分	是，摄像头监控异常	2022-10-24 14:31:43	2022-10-29 15:39:41
23	比特就业课Java方向Java班笔试强训48天day19_11月7日（105Java班）	47.0%	72.5/95	单选:40.0分 编程:32.5分	是，摄像头监控异常	2022-11-06 14:19:57	2022-11-07 11:29:43
24	比特就业课Java方向笔试强训48天day20_11月8日（105Java班）	1.0%	100.0/100	单选:50.0分 编程:50.0分	是，摄像头监控异常|代码抄袭	2022-11-07 09:57:17	2022-11-07 20:19:33

编码能力



题号	正确性	提交次数	做题用时	使用语言	运行时间	占用内存	编程思路	代码规范	成绩排名
编程题1	100%	3	00:50:34	Java	421ms	24680K	良	良	1%
编程题2	75%	5	01:47:16	Java	50ms	10956K	中	良	39%

1 [单选题 | 平均分3.93分 | 66人正确/84人做题 | 用时：<1分 得分： 5.0 / 5.0

设一个有序的单链表中有n个结点，现要求插入一个新结点后使得单链表仍然保持有序，则该操作的时间复杂度（ ）

A O(log2n)
B O(1)
C O(n2)
D O(n)

他的回答： D (正确)

正确答案： D

2 [单选题 | 平均分4.29分 | 73人正确/85人做题 | 用时：<1分 得分： 5.0 / 5.0

一个栈的初始状态为空。首先将元素5，4，3，2，1 依次入栈，然后退栈一次，再将元素A,B,C,D依次入栈，之后将所有元素全部退栈，则所有元素退栈（包括中间退栈的元素）的顺序为（ ）

A 1DCAB2345
B 1DCBA2345
C 54321ABCD
D DCBA12345

他的回答： B (正确)

正确答案： B

3 [单选题 | 平均分4.07分 | 70人正确/86人做题 | 用时：2分 得分： 5.0 / 5.0

设栈S和队列Q的初始状态为空，元素e1，e2，e3，e4，e5，e6依次压入栈S，一个元素出栈后即进入队列Q，若出队列的顺序为e2,e4,e3,e6,e5,e1则栈S的容量要求最小值为（ ）

A 2
B 3
C 4
D 5

他的回答： B (正确)

正确答案： B

4 [单选题 | 平均分3.59分 | 61人正确/85人做题 | 用时：<1分 得分： 5.0 / 5.0

给定下列程序，那么执行printf("%d\n", foo(20, 13));的输出结果是（ ）

```
int foo(int x, int y){
    if (x <= 0 || y <= 0)
        return 1;
    return 3 * foo( x-6, y/2 );
}
```

A 3
B 9
C 27
D 81

他的回答： D (正确)

正确答案： D

参考答案：
解析：foo(20, 13) = 3 * foo(14, 6) = 3 * 3 * foo(8, 3) = 3 * 3 * 3 * foo(2, 1) = 3 * 3 * 3 * 3 * foo(-4, 0) =3 * 3 * 3 * 3 * 1 = 81
答案：D

---

5 [单选题 | 平均分3.33分 | 56人正确/84人做题 | 用时：2分 得分： 0.0 / 5.0

在具有 2n 个结点的完全二叉树中，叶子结点个数为（ ）

A n
B n+1
C n-1
D n/2

偶数个节点的二叉树，必定度为1的节点个数为1
又知道 叶子结点的个数比度为2的节点的个数多1
那么
x + 1 + x - 1 = 2n
x = n

他的回答： C (错误)

正确答案： A

参考答案：
完全二叉树是指除最后一层外，每一层上的结点数均达到最大值，在最后一层上只缺少右边的若干结点。根据完全二叉树性质，如果共 2n 个结点，从根结点开始按层序用自然数 1 ， 2 ，… ， 2n 给结点编号，则编号为 n 的结点左子结点编号为 2n ，因此叶子结点编号为 n+1,n+2, … ,2n 。故叶子结点个数为 n ，本题答案为 A 选项。

---

6 [单选题 | 平均分2.82分 | 48人正确/85人做题 | 用时：<1分 得分： 0.0 / 5.0

下列叙述中错误的是（ ）

A 二叉链表是二叉树的存储结构
B 循环链表是循环队列的存储结构
C 栈是线性结构
D 循环队列是队列的存储结构

循环队列是队列的一种顺序存储结构，也就是我们用的是数组实现的循环队列

他的回答： A (错误)

正确答案： B

参考答案：
循环队列是队列的一种顺序存储结构，用队尾指针 rear 指向队列中的队尾元素，用排头指针 front 指向排头元素的前一个位置。循环链表是用不连续的存储单元存储数据，它有一个表头结点，队头指针指向表头结点，最后一个结点的指针域指向表头结点。二叉链表是树的二叉链表实现方式。栈是一种特殊存取方式的线性表。故本题答案为 B 选项。

---

7 [单选题 | 平均分1.69分 | 29人正确/86人做题 | 用时：3分 得分： 5.0 / 5.0

下述二叉树中，哪一种满足性质：从任一结点出发到根的路径上所经过的结点序列按其关键字有序（ ）

A 二叉排序树
B 哈夫曼树
C AVL树
D 堆

二叉排序树你只能是中序遍历走才会有序，不然其他方式是无序的

堆，无论是大根堆，还是小根堆，你的每个子树都能保证根节点是大于或者小于左右子树的，所以无论你从哪个节点出发，得到的1一定是有序的序列

他的回答： D (正确)

正确答案： D

---

8 [单选题 | 平均分3.69分 | 62人正确/84人做题 | 用时：2分 得分： 5.0 / 5.0

为提高散列（Hash）表的查找效率，可以采取的正确措施是（ ）
Ⅰ．增大装填（载）因子
Ⅱ．设计冲突（碰撞）少的散列函数
Ⅲ．处理冲突（碰撞）时避免产生聚集（堆积）现象

A 仅 I
B 仅 II
C 仅 I、 II
D 仅 II、 III

他的回答： D (正确)

正确答案： D

9 [单选题 | 平均分1.94分 | 33人正确/85人做题 | 用时：6分 得分： 5.0 / 5.0

将整数数组（7-6-3-5-4-1-2）按照堆排序的方式原地进行升序排列，请问在第一轮排序结束之后，数组的顺序是（ ）

A 2-6-3-5-4-1-7
B 6-2-3-5-4-1-7
C 6-5-3-2-4-1-7
D 1-4-7-5-6-3-2

他的回答： C (正确)

正确答案： C

10 [单选题 | 平均分2.08分 | 35人正确/84人做题 | 用时：<1分 得分： 5.0 / 5.0

下列各排序法中，最坏情况下的时间复杂度最低的是（ ）

A 希尔排序
B 快速排序
C 堆排序
D 冒泡排序

他的回答： C (正确)

正确答案： C

参考答案：
堆排序最坏情况时间下的时间复杂度为 O(nlog2n) ；希尔排序最坏情况时间下的时间复杂度为 O(n1.5) ；快速排序、冒泡排序最坏情况时间下的时间复杂度为 O(n2) 。故本题答案为 C 选项。

11 ACM编程题 语言限制 [编程题 | 平均分15.41分 | 53人正确/86人做题 | 提交: 3 次 得分： 25.0 / 25.0

标题：洗牌 | 时间限制：1秒 | 内存限制：32768K | 语言限制： 不限

【洗牌】洗牌在生活中十分常见，现在需要写一个程序模拟洗牌的过程。 现在需要洗2n张牌，从上到下依次是第1张，第2张，第3张一直到第2n张。首先，我们把这2n张牌分成两堆，左手拿着第1张到第n张（上半堆），右手拿着第n+1张到第2n张（下半堆）。接着就开始洗牌的过程，先放下右手的最后一张牌，再放下左手的最后一张牌，接着放下右手的倒数第二张牌，再放下左手的倒数第二张牌，直到最后放下左手的第一张牌。接着把牌合并起来就可以了。 例如有6张牌，最开始牌的序列是1,2,3,4,5,6。首先分成两组，左手拿着1,2,3；右手拿着4,5,6。在洗牌过程中按顺序放下了6,3,5,2,4,1。把这六张牌再次合成一组牌之后，我们按照从上往下的顺序看这组牌，就变成了序列1,4,2,5,3,6。 现在给出一个原始牌组，请输出这副牌洗牌k次之后从上往下的序列。

输入描述：

> 第一行一个数T(T ≤ 100)，表示数据组数。对于每组数据，第一行两个数n,k(1 ≤ n,k ≤ 100)，接下来一行有2n个数a1,a2,...,a2n(1 ≤ ai ≤ 1000000000)。表示原始牌组从上到下的序列。

输出描述：

> 对于每组数据，输出一行，最终的序列。数字之间用空格隔开，不要在行末输出多余的空格。

示例1：

输入

```
3 3 1 1 2 3 4 5 6 3 2 1 2 3 4 5 6 2 2 1 1 1 1
```

输出

```
1 4 2 5 3 6 1 5 4 3 2 6 1 1 1 1
```

代码片段

功能实现	TA的	平均
总通过率	100%	61%
基本测试用例通过率	6/6 (100%)	61%
边缘测试用例通过率	4/4 (100%)	61%

代码提交统计	TA的	平均
使用语言	Java	
做题用时	00:50:34	00:29:50
提交次数	3	1

代码执行统计

答案正确 ：3

代码效率	TA的	参考
运行时间	421ms	1s
占用内存	24680K	32768K

代码规范及可读性

代码规范得分 4.0

Line 2: Wrong lexicographical order for 'java.lang.System' import. Should be before 'java.util.*'. [CustomImportOrder]

Line 3: 'CLASS_DEF' should be separated from previous statement. [EmptyLineSeparator]

Line 6:13: Local variable name 'n' must match pattern '^[a-z][a-z0-9][a-zA-Z0-9]*$'. [LocalVariableName]

Line 9:17: Local variable name 'k' must match pattern '^[a-z][a-z0-9][a-zA-Z0-9]*$'. [LocalVariableName]

Line 18:21: Local variable name 'm' must match pattern '^[a-z][a-z0-9][a-zA-Z0-9]*$'. [LocalVariableName]

他的代码：

做题用时: 50 分钟　　语言：Java　　运行时间：421ms　　占用内存：24680K　　程序状态：答案正确

```
import java.util.*;
import java.lang.System;
public class Main {
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        while(n != 0){
            int num = scan.nextInt();//表示这一手牌有多少张
            int k = scan.nextInt();//表示洗牌的次数
            int[] arr = new int[2*num];//存放牌
            for(int i = 0;i < arr.length;i++){
                arr[i] = scan.nextInt();
            }
            //开始洗牌
            int len = arr.length;
            while(k != 0){
                int[] tmp = new int[len];
                int m = 0;
                for(int i = 0;i < len/2;i++){
                    tmp[m++] = arr[i];
                    tmp[m++] = arr[i+len/2];
                }
                System.arraycopy(tmp,0,arr,0,len);
                k--;
            }
            for(int i = 0;i < len;i++){
                if(i == len - 1){
                    System.out.print(arr[i]);
                }else{
                    System.out.print(arr[i] + " ");
                }
            }
```

```
            System.out.println();
            n--;
        }
    }
}
```

点此或手机扫描二维码查看代码编写过程

12 ACM编程题 语言限制 [编程题 | 平均分11.64分 | 33人正确/84人做题 | 提交: 5 次 得分： 18.75 / 25.0

标题：MP3光标位置 | 时间限制：1秒 | 内存限制：32768K | 语言限制： 不限

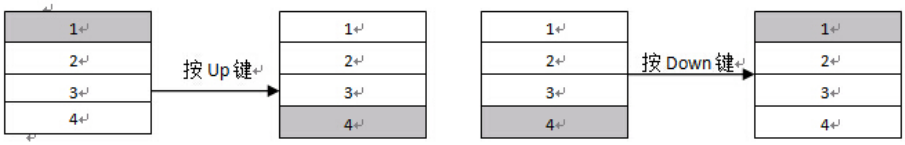
【MP3光标位置】

MP3 Player因为屏幕较小，显示歌曲列表的时候每屏只能显示几首歌曲，用户要通过上下键才能浏览所有的歌曲。为了简化处理，假设每屏只能显示4首歌曲，光标初始的位置为第1首歌。

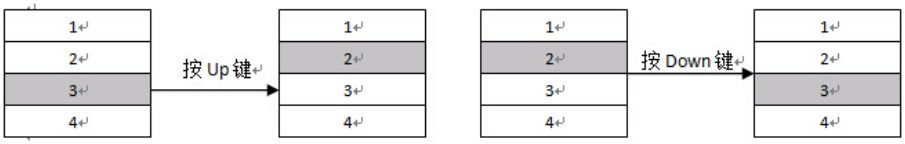
现在要实现通过上下键控制光标移动来浏览歌曲列表，控制逻辑如下：

歌曲总数<=4的时候，不需要翻页，只是挪动光标位置。

光标在第一首歌曲上时，按Up键光标挪到最后一首歌曲；光标在最后一首歌曲时，按Down键光标挪到第一首歌曲。

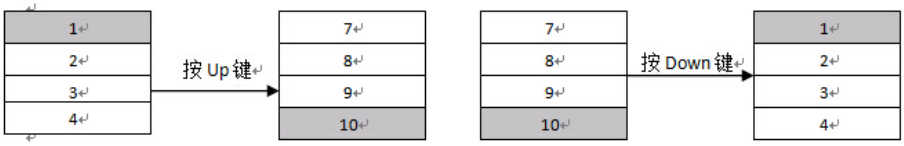


其他情况下用户按Up键，光标挪到上一首歌曲；用户按Down键，光标挪到下一首歌曲。

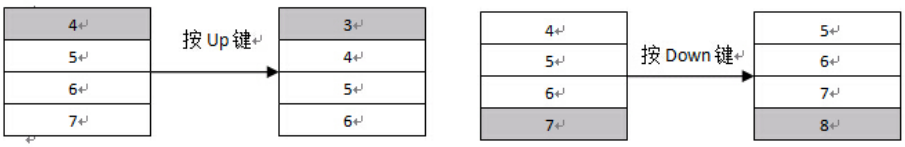


2. 歌曲总数大于4的时候（以一共有10首歌为例）：

特殊翻页：屏幕显示的是第一页（即显示第1－4首）时，光标在第一首歌曲上，用户按Up键后，屏幕要显示最后一页（即显示第7-10首歌），同时光标放到最后一首歌上。同样的，屏幕显示最后一页时，光标在最后一首歌曲上，用户按Down键，屏幕要显示第一页，光标挪到第一首歌上。



一般翻页：屏幕显示的不是第一页时，光标在当前屏幕显示的第一首歌曲时，用户按Up键后，屏幕从当前歌曲的上一首开始显示，光标也挪到上一首歌曲。光标当前屏幕的最后一首歌时的Down键处理也类似。



其他情况，不用翻页，只是挪动光标就行。

数据范围：命令长度 $1 \leq s \leq 100$ ，歌曲数量 $1 \leq n \leq 150$

进阶：时间复杂度： $O(n)$ ，空间复杂度： $O(n)$

输入描述：

> 输入说明：
> 1 输入歌曲数量
> 2 输入命令 U或者D

输出描述：

> 输出说明
> 1 输出当前列表
> 2 输出当前选中歌曲

示例1：

输入

> 10
> UUUU

输出

> 7 8 9 10
> 7

代码片段

功能实现			代码提交统计			代码执行统计
	TA的	平均		TA的	平均	答案错误 ：5
总通过率	75%	46%	使用语言	Java		
基本测试用例通过率	9/12 (75%)	47%	做题用时	01:47:16	00:39:36	
边缘测试用例通过率	6/8 (75%)	45%	提交次数	5	2	

代码效率			代码规范及可读性	
	TA的	参考	代码规范得分	4.75904
运行时间	50ms	1s	Line 2: 'CLASS_DEF' should be separated from previous statement. [EmptyLineSeparator]	
占用内存	10956K	32768K	Line 5:13: Local variable name 'n' must match pattern '^[a-z][a-z0-9][a-zA-Z0-9]*$'. [LocalVariableName]	

他的代码：

做题用时: 107 分钟　语言：Java　运行时间：50ms　占用内存：10956K　程序状态：答案错误

```
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        String command = scan.next();
        int[] song = new int[n];
        for(int i = 0;i < n;i++){
            song[i] = i + 1;
        }
        int left = 0;
        int right = 0;
```

```
int cursor = 0;//光标默认在第一首歌
if(n <= 4){
    right = song.length - 1;
    //这里就只需要在一页内进行上下移动
    for(int i = 0;i < command.length();i++){
        char ch = command.charAt(i);
        if(ch == 'U'){
            if(cursor == 0){
                cursor = song.length - 1;
            }else{
                cursor = cursor - 1;
            }
        }else if(ch == 'D'){
            cursor = (cursor + 1)%song.length;
        }
    }
}else{
    //歌曲数量大于4了
    right = 3;//默认列表是第一页的四首歌曲
    for(int i = 0;i < command.length();i++){
        char ch = command.charAt(i);
        if(ch == 'U'){
            if(cursor == 0){
                cursor = song.length - 1;
                right = song.length - 1;
                if(n <= 8){
                    left = 4;
                }else{
                    left = right - 3;
                }
            }else{
                cursor = cursor - 1;
                if(cursor < left){
                    if(n <= 8){
                        left = 0;
                        right = 3;
                    }else{
                        left--;
                        right--;
                    }
                }
            }
        }else if(ch == 'D'){
            cursor = (cursor + 1)%song.length;
            if(cursor == song.length - 1){
                left = 0;
                right = left + 3;
            }else{
                if(cursor > right){
                    if(n <= 8){
                        left = 4;
                        right = song.length - 1;
                    }else{
                        left++;
                        right++;
                    }
                }
            }
        }
    }
}
```

```
        //left ,right之间就是要输出的歌曲列表
        for(int i = left;i <= right;i++){
            System.out.print(song[i] + " ");
        }
        System.out.println();
        System.out.println(song[cursor]);

    }
}
```

点此或手机扫描二维码查看代码编写过程

监控截图