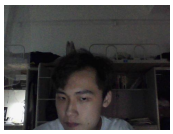
[

每日一题day21_6月12日测评结果

考生信息

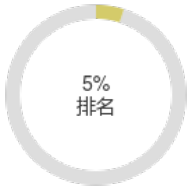


张博翔

考号：1675 | 学校：陕西科技大学 | 邮箱：1761607418@qq.com | 职位：43班 |

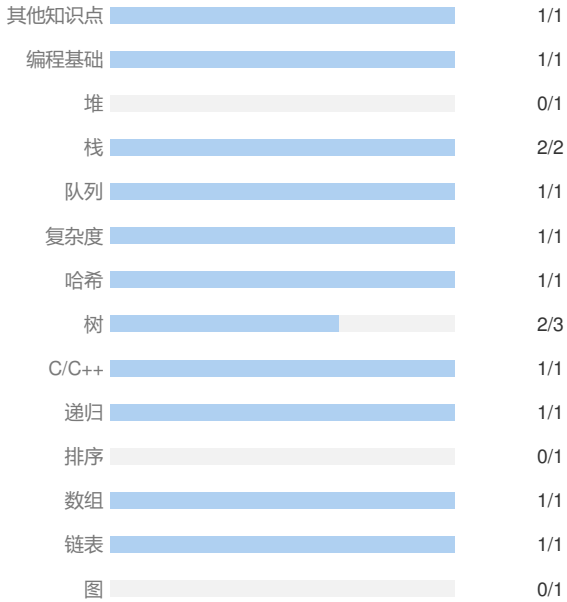
参考区域: 陕西省西安市（111.114.0.1） | 做题用时：03:20:14(2019-06-11 22:30:09 - 2019-06-12 01:50:35)

考生成绩



题型	得分	正确题数	排名	用时	是否阅卷
单选	35.0	7	21	00:37:21	--
编程	50.0	2	1	02:38:42	--

知识点技能图谱

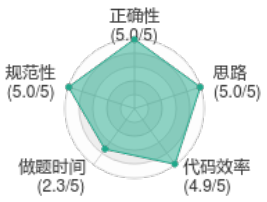


知识点	得分	正确题数
其他知识点	25.0	1
编程基础	5.0	1
堆	0.0	0
栈	10.0	2
队列	5.0	1
复杂度	5.0	1
哈希	5.0	1
树	10.0	2
C/C++	5.0	1
递归	5.0	1
排序	0.0	0
数组	25.0	1
链表	5.0	1
图	0.0	0

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	笔试时间
1	每日一题day1_5月20日	4.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-19 16:08:49
2	每日一题day02_5月21日	51.0%	60.0/100	单选:30.0分 编程:30.0分	否	2019-05-20 17:40:56
3	每日一题day03_5月22日	4.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-21 12:59:15
4	每日一题day04_5月23日	8.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-22 10:40:18
5	每日一题day05_5月24日	29.0%	75.0/100	单选:25.0分 编程:50.0分	否	2019-05-22 20:39:16
6	每日一题day06_5月25日	10.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-24 10:41:21
7	每日一题day07_5月27日	6.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-26 15:09:41
8	每日一题day08_5月28日	18.0%	90.0/100	单选:40.0分 编程:50.0分	否	2019-05-27 14:06:31
9	每日一题day09_5月29日	13.0%	90.0/100	单选:40.0分 编程:50.0分	否	2019-05-28 15:11:53
10	每日一题day10_5月30日	6.0%	90.0/100	单选:40.0分 编程:50.0分	否	2019-05-29 11:32:20
11	每日一题day11_5月31日	14.0%	80.0/100	单选:30.0分 编程:50.0分	否	2019-05-30 11:22:09
12	每日一题day12_6月1日	31.0%	75.0/100	单选:25.0分 编程:50.0分	否	2019-05-31 10:33:31
13	每日一题day13_6月3日	9.0%	85.0/100	单选:35.0分 编程:50.0分	否	2019-06-02 15:20:12
14	每日一题day14_6月4日	41.0%	50.0/100	单选:25.0分 编程:25.0分	否	2019-06-03 11:53:13
15	每日一题day15_6月5日	6.0%	85.0/100	单选:35.0分 编程:50.0分	否	2019-06-04 12:02:48
16	每日一题day16_6月6日	6.0%	80.0/100	单选:30.0分 编程:50.0分	否	2019-06-05 16:32:52
17	每日一题day17_6月7日	4.0%	90.0/100	单选:40.0分 编程:50.0分	否	2019-06-06 10:57:01
18	每日一题day18_6月8日	16.0%	90.0/100	单选:40.0分 编程:50.0分	否	2019-06-07 15:47:43
19	每日一题day19_6月10日	3.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-06-09 11:55:46
20	每日一题day20_6月11日	4.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-06-10 14:43:47

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%	1	00:39:03	C++	20ms	616K	优	优	1%
编程题2	100%	3	01:59:39	C++	4ms	488K	优	优	1%

1 [平均分3.9分 | 75人正确/95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 [平均分4.4分 | 83人正确/94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 [平均分4.4分 | 83人正确/94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 [平均分3.6分 | 67人正确/94人做题 | 用时：5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 [平均分3.0分 | 56人正确/94人做题 | 用时：3分 得分：5.0 / 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

他的回答： A (正确)

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 [平均分2.0分 | 34人正确/83人做题 | 用时：13分 得分：5.0 / 5.0

有权值分别为11，8，6，2，5的叶子结点生成一棵哈夫曼树，它的带权路径长度为_____。

A 24

B 71

C 48

D 53

他的回答： B (正确)

正确答案： B

7 [平均分1.9分 | 35人正确/91人做题 | 用时：<1分 得分：0.0 / 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

他的回答： A (错误)

正确答案： D

8 [平均分3.6分 | 62人正确/87人做题 | 用时：<1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 [平均分2.4分 | 43人正确/90人做题 | 用时：8分 得分：0.0 / 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 1-5-3-2-4-6-7

E 5-4-3-2-1-6-7

F 5-1-3-2-4-6-7

他的回答： A (错误)

正确答案： C

10 [平均分1.7分 | 31人正确/90人做题 | 用时：<1分 得分：0.0 / 5.0

要连通具有 n 个顶点的有向图，最少需要（ ）条边。

A n+l

B n-l

C 2n

D n

他的回答： B (错误)

正确答案： D

---

11 [平均分18.8分 | 45人正确/60人做题 | 提交: 1 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			代码提交统计			代码执行统计
	TA的	平均		TA的	平均	答案正确 ：1
总通过率	100%	75%	使用语言	C++		
基本测试用例通过率	6/6 (100%)	75%	做题用时	00:39:03	01:04:44	
边缘测试用例通过率	4/4 (100%)	75%	提交次数	1	6	

代码效率			代码规范及可读性
	TA的	参考	代码规范得分 5.0
运行时间	20ms	1s	
占用内存	616K	32768K	

他的代码：

做题用时: 39 分钟 语言：C++ 运行时间：20ms 占用内存：616K 程序状态：答案正确

```
#include<iostream>
#include<vector>
using namespace std;

int main()
```

```
{
    int T, n, k;
    cin >> T;
    while (T--)
    {
        cin >> n >> k;
        int num = 2 * n;
        vector<int> table(num);
        for(int i = 0; i < num; ++i)
            cin >> table[i];
        while (k--)
        {
            vector<int> n1(table.begin(), table.end());
            for (int i = 0; i < n; ++i)
            {
                table[2 * i] = n1[i];
                table[2 * i + 1] = n1[i + n];
            }
        }
        for(int i = 0; i < num - 1; ++i)
            cout << table[i] << " ";
        cout << table[num - 1] << endl;
    }
    return 0;
}
```

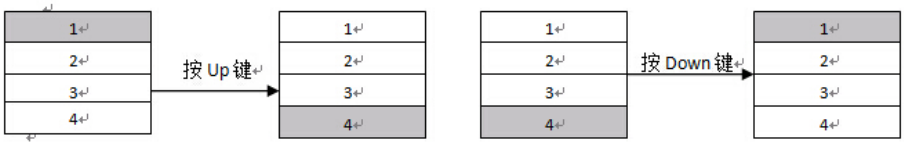
12 [平均分17.7分 | 37人正确/60人做题 | 提交: 3 次 得分：25.0 / 25.0

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

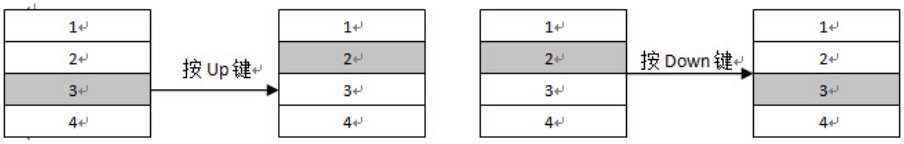
【MP3光标位置】

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。

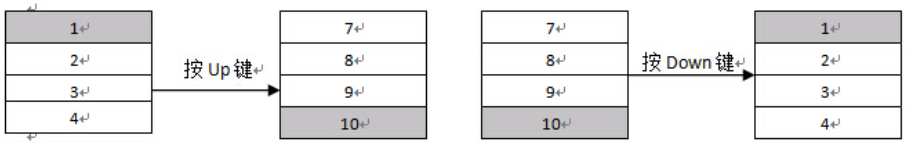


其他情况下用户按Up键，光标挪到上一首歌曲；用户按Down键，光标挪到下一首歌曲。

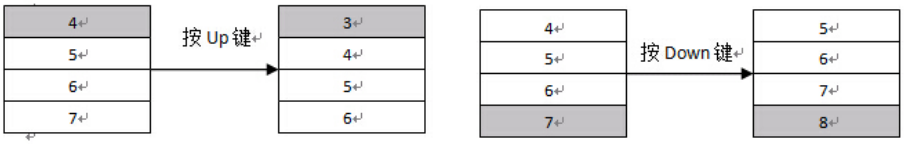


2. 歌曲总数大于4的时候（以一共有10首歌为例）：

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。

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	段错误 ：2
总通过率	100%	70%	使用语言	C++		答案正确 ：1
基本测试用例通过率	6/6 (100%)	71%	做题用时	01:59:39	01:15:03	
边缘测试用例通过率	4/4 (100%)	70%	提交次数	3	6	

代码效率			代码规范及可读性
	TA的	参考	代码规范得分 5.0
运行时间	4ms	1s	
占用内存	488K	32768K	

他的代码：

做题用时: 119 分钟　　语言：C++　　运行时间：4ms　　占用内存：488K　　程序状态：答案正确

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
int main()
{
    int num = 0;
    string str;
    while(scanf("%d", &num) == 1)
    {
        cin >> str;
        int head = 1;//当前视图的第一个
        int pos = 0;//当前视图中所选中的下标
        if(num <= 4)
        {
            for(int i = 0; i < str.size(); i++)
            {
                if(str[i] == 'U')
                {
                    if(pos == 0)
                    {
```

```
                pos = num - 1;
            }
            else
            {
                pos--;
            }
        }
        if(str[i] == 'D')
        {
            if(pos == num - 1)
            {
                pos = 0;
            }
            else
            {
                pos++;
            }
        }
    }
}
else
{
    for(int i = 0; i < str.size(); i++)
    {
        if(str[i] == 'U')
        {
            if(head == 1 && pos == 0)
            {
                head = num - 3;
                pos = 3;
            }
            else
            {
                if(pos > 0)
                {
                    pos--;
                }
                else
                {
                    head--;
                }
            }
        }
        else
        {
            if(head == num - 3 && pos == 3)
            {
                head = 1;
                pos = 0;
            }
            else
            {
                if(pos < 3)
                {
                    pos++;
                }
                else
                {
                    head++;
                }
            }
```

```
            }
        }
    }
    vector<int> output;
    if(num <= 4)
    {
        for(int i = head; i <= num; i++)
        {
            output.push_back(i);
        }
    }
    else
    {
        for(int i = head; i < head + 4; i++)
        {
            output.push_back(i);
        }
    }
    for(int i = 0; i < output.size() - 1; i++)
    {
        cout << output[i] << " ";
    }
    cout << output[output.size() - 1] << endl;
    cout << (head + pos) << endl;
  }
}
```