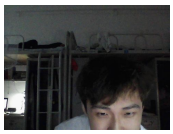
[

每日一题day23_6月14日测评结果

考生信息

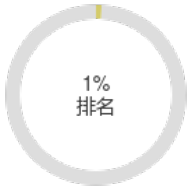


张博翔

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

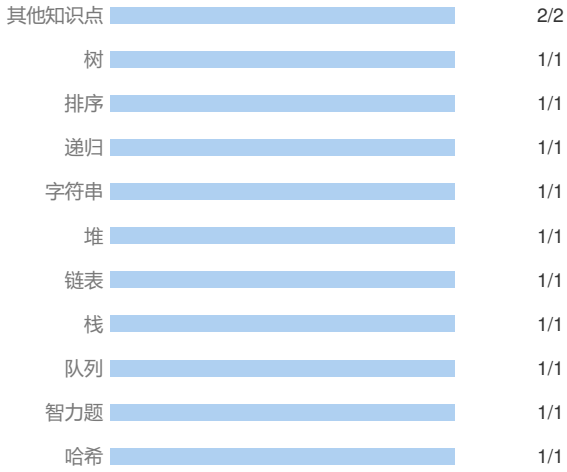
参考区域: 陕西省西安市（111.114.0.2） | 做题用时：03:11:58(2019-06-13 22:56:26 - 2019-06-14 02:08:28)

考生成绩



题型	得分	正确题数	排名	用时	是否阅卷
单选	50.0	10	1	01:32:19	--
编程	50.0	2	1	01:29:43	--

知识点技能图谱



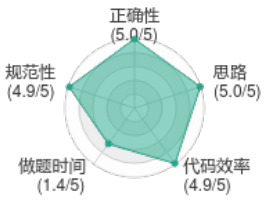
知识点	得分	正确题数
其他知识点	30.0	2
树	5.0	1
排序	5.0	1
递归	5.0	1
字符串	25.0	1
堆	5.0	1
链表	5.0	1
栈	5.0	1
队列	5.0	1
智力题	5.0	1
哈希	5.0	1

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	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

序号	试卷名称	排名	总得分	得分详情	作弊嫌疑	笔试时间
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
21	每日一题day21_6月12日	5.0%	85.0/100	单选:35.0分 编程:50.0分	否	2019-06-11 11:37:10
22	每日一题day22_6月13日	6.0%	90.0/100	单选:40.0分 编程:50.0分	否	2019-06-12 11:18:48

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%	7	00:36:07	C++	9ms	476K	优	优	1%
编程题2	100%	5	00:53:36	C++	8ms	736K	良	良	1%

1 [平均分4.1分 | 65人正确/79人做题 | 用时：4分 得分：5.0 / 5.0

在双向循环链表中，在p指针所指的节点后插入一个指针q所指向的新节点，修改指针的操作是____。

A p->next=q;q->prior=p;p->next->prior=q;q->next=q;

B p->next=q;p->next->prior=q;q->prior=p;q->next=p->next;

C q->prior=p;q->next=p->next;p->next->prior=q;p->next=q;

D q->next=p->next;q->prior=;p->next=q;p->next=q;

他的回答： C (正确)

正确答案： C

2 [平均分3.1分 | 49人正确/78人做题 | 用时：14分 得分：5.0 / 5.0

采用递归方式对顺序表进行快速排序，下列关于递归次数的叙述中，正确的是（ ）

A 递归次数与初始数据的排列次序无关

B 每次划分后，先处理较长的分区可以减少递归次数

C 每次划分后，先处理较短的分区可以减少递归次数

D 递归次数与每次划分后得到的分区处理顺序无关

他的回答： D (正确)

正确答案： D

3 [平均分2.5分 | 38人正确/76人做题 | 用时：5分 得分：5.0 / 5.0

若用一个大小为6的数组来实现循环队列,且当前rear和front的值分别为0和3,当从队列中删除一个元素,再加入两个元素后,rear和front的值分别为多少?()

A 1和5

B 2和4

C 4和2

D 5和1

他的回答： B (正确)

正确答案： B

4 [平均分4.4分 | 68人正确/77人做题 | 用时：5分 得分：5.0 / 5.0

表达式a*(b+c)-d的后缀表达式是()

A abcd*+-

B abc+*d-

C abc*+d-

D -+*abcd

他的回答： B (正确)

正确答案： B

5 [平均分3.4分 | 51人正确/74人做题 | 用时：30分 得分：5.0 / 5.0

一棵完全二叉树第六层有9个叶结点（根为第一层），则结点个数最多有（ ）

A 112

B 111

C 107

D 109

他的回答： D (正确)

正确答案： D

6 [平均分4.6分 | 72人正确/79人做题 | 用时：2分 得分：5.0 / 5.0

两个人两个小时能组装两辆自行车,要在6小时内组装12辆自行车,需要多少人?

A 2

B 3

C 4

D 5

他的回答： C (正确)

正确答案： C

7 [平均分1.6分 | 23人正确/71人做题 | 用时：5分 得分：5.0 / 5.0

已知一个线性表（38，25，74，63，52，48），假定采用散列函数h（key) = key%7 计算散列地址，并散列存储在散列表A【0....6】中，若采用线性探测方法解决冲突，则在该散列表上进行等概率成功查找的平均查找长度为

A 1.5
B 1.7
C 2.0
D 2.3

他的回答： C (正确)

正确答案： C

参考答案：

依次进行取模运算求出哈希地址：

A	0	1	2	3	4	5	6
记录	63	48		38	25	74	52
查找次数	1	3		1	1	2	4

74 应该放在下标为 4 的位置，由于 25 已经放在这个地方，所以 74 往后移动，放在了下标为 5 的位置上了。

由于是等概率查找，所以结果为：1/6*（1+3+1+1+2+4）= 2.0

8 [平均分2.8分 | 43人正确/78人做题 | 用时：14分 得分：5.0 / 5.0

以30为基准,设一组初始记录关键字序列为 (30,15,40,28,50,10,70), 则第一趟快速排序结果为（ ）

A 10，28，15，30，50，40，70
B 10，15，28，30,50，40，70
C 10，28，15，30，40，50，70
D 10，15，28，30，40，50，70

他的回答： B (正确)

正确答案： B

9 [平均分4.0分 | 62人正确/77人做题 | 用时：3分 得分：5.0 / 5.0

一棵二叉树的先序遍历为EFHIGJK，中序遍历为HFIEJKG，则后序遍历为（ ）

A HIFJKGE
B FHIJKGE
C HIFGJKE
D HIFKJGE

他的回答： D (正确)

正确答案： D

10 [平均分4.2分 | 65人正确/77人做题 | 用时：4分 得分：5.0 / 5.0

已知关键字序列5,8,12,19,28,20,15,22是最小堆，插入关键字3，调整后得到的最小堆是()

A 3,8,12,5,20,15,22,28,19
B 3,5,12,19,20,15,22,8,28
C 3,12,5,8,28,20,15,22,19
D 3,5,12,8,28,20,15,22,19

他的回答： D (正确)

正确答案： D

11 [平均分22.9分 | 64人正确/70人做题 | 提交: 7 次 得分：25.0 / 25.0

标题：微信红包 | 时间限制：3秒 | 内存限制：32768K | 语言限制： [Python, C++, C#, Java]

【微信红包】

春节期间小明使用微信收到很多个红包，非常开心。在查看领取红包记录时发现，某个红包金额出现的次数超过了红包总数的一半。请帮小明找到该红包金额。写出具体算法思路和代码实现，要求算法尽可能高效。

给定一个红包的金额数组**gifts**及它的大小**n**，请返回所求红包的金额。

若没有金额超过总数的一半，返回0。

测试样例：

> [1,2,3,2,2],5

> 返回：2

输入描述：

输出描述：

代码片段

功能实现		
	TA的	平均
总通过率	100%	91%
基本测试用例通过率	1/1 (100%)	91%

代码提交统计		
	TA的	平均
使用语言	C++	
做题用时	00:36:07	00:19:24
提交次数	7	4

代码执行统计
答案错误 ：2
编译错误 ：4
答案正确 ：1

代码效率		
	TA的	参考
运行时间	9ms	3s
占用内存	476K	32768K

代码规范及可读性
代码规范得分 5.0

他的代码：

做题用时: 36 分钟 语言：C++ 运行时间：9ms 占用内存：476K 程序状态：答案正确

```
#include <iostream>
#include <vector>
using namespace std;
class Gift {
public:
    int getValue(vector<int> gifts, int n) {
        // write code here
        for(int i = 0; i < gifts.size(); i++)
        {
            int count = 0;
            for(int j = i; j < gifts.size(); j++)
            {
                if(gifts[j] == gifts[i])
                {
                    cout << gifts[i] << " " << gifts[j] << endl;
                    count++;
                }
            }
            if(count > (n / 2))
            {
                //cout << i << " " << count << endl;
                return gifts[i];
                cout << count << endl;
```

```
            }
            else
            {
                int temp = gifts[i];
                i--;
                vector<int>::iterator it = gifts.begin();
                while(it != gifts.end())
                {
                    if(*it == temp)
                    {
                        cout << *it << endl;
                        it = gifts.erase(it);
                    }
                    else
                    {
                        ++it;
                    }
                }
            }
        }
        return 0;
    }
};
```

12 [平均分13.6分 | 24人正确/44人做题 | 提交: 5 次 得分：25.0 / 25.0

标题：计算字符串的距离 | 时间限制：1秒 | 内存限制：32768K | 语言限制： 不限

【计算字符串的距离】

Levenshtein 距离，又称编辑距离，指的是两个字符串之间，由一个转换成另一个所需的最少编辑操作次数。许可的编辑操作包括将一个字符替换成另一个字符，插入一个字符，删除一个字符。编辑距离的算法是首先由俄国科学家Levenshtein提出的，故又叫Levenshtein Distance。
Ex：
字符串A:abcdefg
字符串B: abcdef
通过增加或是删掉字符”g”的方式达到目的。这两种方案都需要一次操作。把这个操作所需要的次数定义为两个字符串的距离。
要求：
给定任意两个字符串，写出一个算法计算它们的编辑距离。

请实现如下接口

```
/* 功能：计算两个字符串的距离
 * 输入： 字符串A和字符串B
 * 输出：无
 * 返回：如果成功计算出字符串的距离，否则返回-1
 */
    public static int calStringDistance (String charA, String charB)
  {
      return 0;
  }
```

输入描述：

> 输入两个字符串

输出描述：

> 得到计算结果

示例1：

输入

> abcdefg
> abcdef

输出

```
1
```

代码片段

功能实现	TA的	平均
总通过率	100%	54%
基本测试用例通过率	6/6 (100%)	54%
边缘测试用例通过率	4/4 (100%)	54%

代码提交统计	TA的	平均
使用语言	C++	
做题用时	00:53:36	00:53:59
提交次数	5	4

代码执行统计

答案错误 ：4
答案正确 ：1

代码效率	TA的	参考
运行时间	8ms	1s
占用内存	736K	32768K

代码规范及可读性

代码规范得分 4.8
Line 33: Add #include for min [build/include_what_you_use] [4]

他的代码：

做题用时: 53 分钟　　语言：C++　　运行时间：8ms　　占用内存：736K　　程序状态：答案正确

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
int main()
{
    string str1, str2;
    while(cin >> str1 >> str2)
    {
        int m = str1.size();
        int n = str2.size();
        //cout <<m << n << endl;
        vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));
        for(int i = 0; i <= n; i++)
        {
            dp[0][i] = 1 * i;
        }
        for(int i = 0; i <= m; i++)
        {
            dp[i][0] = 1 * i;
        }
        for(int i = 1; i <= m; i++)
        {
            for(int j = 1; j <= n; j++)
            {
                int case1 = dp[i - 1][j - 1];//完全相等
                int case2 = dp[i][j - 1] + 1;//增加一个
                int case3 = dp[i - 1][j] + 1;//删除一个
                if(str1[i - 1] != str2[j - 1])
                {
                    case1 += 1;
                }
                dp[i][j] = min(min(case1, case2), case3);
            }
        }
        cout << dp[m][n] << endl;
    }
```

```
}
```