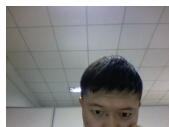


Java方向每日一题day19_1月4日-任栋-测评结果

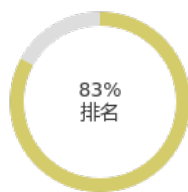
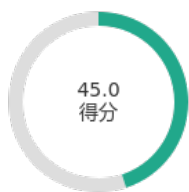
考生信息



任栋

投递编号：2 | 学校：西安理工大学 | 邮箱：1104580363@qq.com | 职位：2020大四春招冲刺班 |
参考区域：陕西省西安市（221.11.20.102） | 做题用时：03:14:45(2021-01-05 17:47:19开始答题，21:02:14交卷) |
作答设备：PC | 已同意诚信声明和隐私协议

考生成绩



题型	得分	正确题数	排名	用时	是否阅卷
单选	20.0	4	44	00:14:53	--
编程	25.0	1	22	02:43:45	--

知识点技能图谱

递归	0/1
树	2/2
排序	1/2
字符串	1/1
查找	1/1
堆	0/1
链表	1/2
栈	0/1
队列	0/1
高级结构	1/1
哈希	0/1
图	0/1

知识点	得分	正确题数
递归	0.0	0
树	10.0	2
排序	5.0	1
字符串	25.0	1
查找	5.0	1
堆	0.0	0
链表	5.0	1
栈	0.0	0
队列	0.0	0
高级结构	25.0	1
哈希	0.0	0
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	安排笔试时间	交卷时间
----	------	----	-----	------	------	--------	------

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	大四春招冲刺班JavaSE考试	31.0%	26.0/60	单选:26.0分	否	2020-10-28 17:33:16	2020-10-29 10:30:13
2	大四春招冲刺班数据结构考试	77.0%	22.0/60	单选:22.0分 编程:0.0分	否	2020-11-23 11:55:15	2020-11-24 10:40:01
3	Java方向每日一题day02_11月24日	70.0%	55.0/100	单选:30.0分 编程:25.0分	否	2020-11-23 12:10:19	2020-11-25 15:36:05
4	Java方向每日一题day03_11月25日	79.0%	55.0/100	单选:30.0分 编程:25.0分	否	2020-11-24 15:19:25	2020-11-24 22:21:11
5	Java方向每日一题day04_11月26日	61.0%	85.0/100	单选:35.0分 编程:50.0分	否	2020-11-25 14:58:20	2020-11-25 23:17:45
6	Java方向每日一题day05_11月27日	16.0%	85.0/100	单选:35.0分 编程:50.0分	否	2020-11-25 15:54:38	2020-11-26 23:32:02
7	Java方向每日一题day06_11月28日	43.0%	77.5/100	单选:30.0分 编程:47.5分	否	2020-11-27 14:19:26	2020-11-27 22:53:24
8	Java方向每日一题day07_11月30日	1.0%	100.0/100	单选:50.0分 编程:50.0分	否	2020-11-29 13:58:20	2020-11-30 22:17:40
9	Java方向每日一题day08_12月1日	28.0%	90.0/100	单选:40.0分 编程:50.0分	是，代码抄袭	2020-11-30 10:48:03	2020-12-01 22:29:01
10	Java方向每日一题day09_12月2日	86.0%	50.0/100	单选:35.0分 编程:15.0分	否	2020-12-01 10:43:40	2020-12-02 21:12:07
11	Java方向每日一题day10_12月3日	44.0%	75.0/100	单选:25.0分 编程:50.0分	否	2020-12-02 12:27:01	2020-12-04 09:14:04
12	Java方向每日一题day11_12月4日	72.0%	57.14/100	单选:25.0分 编程:32.14分	否	2020-12-03 10:46:54	2020-12-04 11:05:57
13	Java方向每日一题day12_12月5日	66.0%	60.0/100	单选:35.0分 编程:25.0分	否	2020-12-04 10:43:45	2020-12-05 21:50:32
14	每日一题Java方向day13_12月7日	75.0%	48.57/100	单选:40.0分 编程:8.57分	否	2020-12-05 10:31:45	2020-12-07 22:58:00
15	每日一题Java方向day14_12月8日	62.0%	65.0/100	单选:40.0分 编程:25.0分	否	2020-12-07 12:07:00	2020-12-08 16:49:09
16	每日一题Java方向day16_12月10日	34.0%	75.0/100	单选:25.0分 编程:50.0分	否	2020-12-09 10:58:00	2020-12-12 16:11:42
17	每日一题Java方向day17_12月11日	81.0%	45.0/100	单选:45.0分	否	2020-12-09 15:13:19	2020-12-13 16:22:36

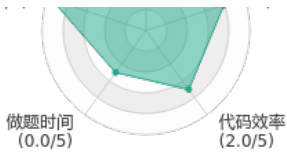
编码能力

规范性
(4.5/5)

正确性
(2.5/5)

思路
(2.9/5)

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%	2	01:04:24	Java	74ms	11676K			2%



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2	0%	8	01:39:21	Java	0ms	0K			-

1 [平均分2.1分 | 19人正确/45人做题 | 用时：<1分 | 得分：5.0 / 5.0]

下列关于线性链表的叙述中，正确的是（ ）。

- A 各数据结点的存储空间可以不连续，但它们的存储顺序与逻辑顺序必须一致
- B 各数据结点的存储顺序与逻辑顺序可以不一致，但它们的存储空间必须连续
- C 进行插入与删除时，不需要移动表中的元素
- D 以上说法均不正确

他的回答： C (正确)

正确答案： C

参考答案：

一般来说，在线性表的链式存储结构中，各数据结点的存储序号是不连续的，并且各结点在存储空间中的位置关系与逻辑关系也不一致。线性链表中数据的插入和删除都不需要移动表中的元素，只需改变结点的指针域即可。

2 [平均分4.3分 | 40人正确/47人做题 | 用时：<1分 | 得分：0.0 / 5.0]

一个栈的初始状态为空。现将元素 1,2,3,A,B,C 依次入栈，然后再依次出栈，则元素出栈的顺序是（ ）

- A 1,2,3,A,B,C
- B C,B,A,1,2,3
- C C,B,A,3,2,1
- D 1,2,3,C,B,A

他的回答： B (错误)

正确答案： C

参考答案：

栈的修改是按后进先出的原则进行的，所以顺序应与入栈顺序相反，故选 C。

3 [平均分2.7分 | 25人正确/47人做题 | 用时：<1分 | 得分：0.0 / 5.0]

用不带头结点的单链表存储队列,其队头指针指向队头结点,队尾指针指向队尾结点,则在进行出队操作时()

- A 仅修改队头指针
- B 仅修改队尾指针
- C 队头、队尾指针都可能要修改
- D 队头、队尾指针都要修改

他的回答： A (错误)

正确答案： C

4 [平均分3.8分 | 36人正确/47人做题 | 用时：<1分 | 得分：0.0 / 5.0]

递归函数最终会结束，那么这个函数一定？

- A 使用了局部变量
- B 有一个分支不调用自身
- C 使用了全局变量或者使用了一个或多个参数
- D 没有循环调用

他的回答： D (错误)

正确答案： B

参考答案：

直接排除AD，注意力集中在B和C。

B肯定是对的，只有一次循环满足某个条件，不调用自己就返回，递归才会一层一层向上返回。

那么C呢，想一下，全局变量和参数确实可以用来控制递归的结束与否。

该不该选C呢？再仔细看一下题目（说实话，我很讨厌这种文字游戏），“这个函数一定...” ，所以，问题集中在，是否是一定会使用这两种方式呢？显然不是的。

除了C中提到的两种情况外，还有如下控制递归的方式：

1. 局部静态变量是可以控制递归函数最终结束的 2. 可能通过异常来控制递归的结束。 3. 可以利用BIOS或OS的一些数据或一些标准库的全局值来控制递归过程的终止。 4. 可以把一些数据写入到BIOS或OS的系统数据区，也可以把数据写入到一个文件中，以此来控制递归函数的终止。

所以，答案为B

5 [平均分4.3分 | 40人正确/46人做题 | 用时：3分 | 得分：5.0 / 5.0]

已知二叉树后序遍历序列是bfegcd，中序遍历序列是badefcg，它的前序遍历序列是：

- A abcdefg
- B abdcefg
- C adbcfeg
- D abecdffg

他的回答：B (正确)

正确答案：B

参考答案：

分析：很有代表性的一道题目，去年参加微软笔试的时候也有类似的题目。后序遍历中的最后一个元素是根节点，a，然后查找中序中a的位置，把中序遍历分成b a defcg，易知左子树为b，右子树为defcg，再递归求解，可画出原始二叉树，故知前序遍历序列为B。

6 [平均分4.2分 | 39人正确/46人做题 | 用时：<1分 | 得分：5.0 / 5.0]

某完全二叉树按层次输出（同一层从左到右）的序列为ABCDEFGH。该完全二叉树的前序序列为（ ）

- A ABDHECFG
- B ABCDEFGH
- C HDBEAFCG
- D HDEBFGCA

他的回答：A (正确)

正确答案：A

参考答案：

前序遍历：访问根结点在访问左子树和访问右子树之前。即先访问根结点，然后遍历左子树，最后遍历右子树；并且在遍历左子树和右子树时，仍然先访问根结点，然后遍历左子树，最后遍历右子树。中序遍历：访问根结点在访问左子树和访问右子树两者之间。即先遍历左子树，然后访问根结点，最后遍历右子树。并且在遍历左子树和右子树时，仍然首先遍历左子树，然后访问根结点，最后遍历右子树。后序遍历：访问根结点在访问左子树和访问右子树之后。即首先遍历左子树，然后遍历右子树，最后访问根结点；并且在遍历左子树和右子树时，仍然首先遍历左子树，然后遍历右子树，最后访问根结点。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。因此此完全二叉树可能的形状为：则前序遍历序列为：ABDHECFG。故本题答案为A选项。

7 [平均分3.7分 | 34人正确/46人做题 | 用时：5分 | 得分：0.0 / 5.0]

以下序列不是堆的是()

- A (100,85,98,77,80,60,82,40,20,10,66)
- B (100,98,85,82,80,77,66,60,40,20,10)
- C (10,20,40,60,66,77,80,82,85,98,100)
- D (100,85,40,77,80,60,66,98,82,10,20)

他的回答：C (错误)

正确答案：D

8 [平均分2.4分 | 22人正确/45人做题 | 用时：2分 | 得分：0.0 / 5.0]

设有一组记录的关键字为(19,14,23,1,68,20,84,27,55,11,10,79),用链地址法构造哈希表,哈希函数为 $H(\text{key})=\text{key} \bmod 13$,哈希地址为1的链中有()个记录

- A 1
- B 2
- C 3
- D 4

他的回答：C (错误)

正确答案：D

9 [平均分4.8分 | 44人正确/46人做题 | 用时：<1分 | 得分：5.0 / 5.0

假设你只有100Mb的内存，需要对1Gb的数据进行排序，最合适的算法是？

- A 归并排序
- B 插入排序
- C 快速排序
- D 冒泡排序

他的回答：A (正确)

正确答案：A

10 [平均分1.8分 | 17人正确/46人做题 | 用时：<1分 | 得分：0.0 / 5.0

下列哪种图的邻接矩阵是对称矩阵（ ）。

- A 有向图
- B 无向图
- C AOV图
- D AOE图

他的回答：A (错误)

正确答案：B

11 [完善核心代码 | 语言限制] [平均分22.9分 | 43人正确/47人做题 | 提交: 2 次 | 得分：25.0 / 25.0

标题：子串判断 | 时间限制：3秒 | 内存限制：32768K | 语言限制：[Python, C++, C#, Java]

【子串判断】

现有一个小写英文字母组成的字符串s和一个包含较短小写英文字符串的数组p，请设计一个高效算法，对于p中的每一个较短字符串，判断其是否为s的子串。给定一个string数组p和它的大小n，同时给定string s，为母串，请返回一个bool数组，每个元素代表p中的对应字符串是否为s的子串。保证p中的串长度小于等于8，且p中的串的个数小于等于500，同时保证s的长度小于等于1000。

测试样例：

["a","b","c","d"],4,"abc"

返回：[true,true,true,false]

输入描述：

输出描述：

代码片段

功能实现

代码提交统计

代码执行统计

总通过率 100% 91%
基本测试用例通过率 1/1 (100%) 91%

使用语言 Java
做题用时 01:04:24 00:22:54
提交次数 2 3

编译错误：1
答案正确：1

代码效率

代码规范及可读性

运行时间 74ms 3s
占用内存 11676K 32768K

代码规范得分 4.4
Line 4:41: Parameter name 'p' must match pattern '^[a-z][a-z0-9][a-zA-Z0-9]*\$'. [ParameterName]

Line 4:48: Parameter name 'n' must match pattern '^[a-z][a-z0-9][a-zA-Z0-9]*\$'. [ParameterName]

Line 4:58: Parameter name 's' must match pattern '^[a-z][a-z0-9][a-zA-Z0-9]*\$'. [ParameterName]

他的代码：

做题用时: 64 分钟

语言 : Java

运行时间 : 74ms

占用内存 : 11676K

程序状态 : 答案正确

```
import java.util.*;

public class Substr {
    public boolean[] chkSubStr(String[] p, int n, String s) {
        // write code here
        boolean[] tmp = new boolean[n];
        HashMap<String, Boolean> hashMap = new HashMap<>();
        for (int i = 0; i < n; i++) {
            hashMap.put(p[i], false);
        }
        for (String str: hashMap.keySet()) {
            if (s.contains(str)) {
                hashMap.put(str, true);
            }
        }
        for (int i = 0; i < n; i++) {
            tmp[i] = hashMap.get(p[i]);
        }

        return tmp;
    }
}
```



[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 [平均分16.2分 | 21人正确/33人做题 | 提交: 8 次] 得分: 0.0 / 25.0

标题：成绩排序 | 时间限制：1秒 | 内存限制：32768K | 语言限制：不限

【成绩排序】

查找和排序

题目：输入任意（用户，成绩）序列，可以获得成绩从高到低或从低到高的排列,相同成绩都按先录入排列在前的规则处理。

例示：

```
jack    70
peter   96
Tom     70
smith   67
从高到低 成绩
peter   96
jack    70
Tom     70
```

smith 67
从低到高
smith 67
Tom 70
jack 70
peter 96

输入描述：

输入多行，先输入要排序的人的个数，然后分别输入他们的名字和成绩，以一个空格隔开

输出描述：

按照指定方式输出名字和成绩，名字和成绩之间以一个空格隔开

示例1：

输入

3
0
fang 90
yang 50
ning 70

输出

fang 90
ning 70
yang 50

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	编译错误：8	
总通过率	0%	64%	使用语言	Java			
基本测试用例通过率	0/6 (0%)	65%	做题用时	01:39:21	00:35:33		
边缘测试用例通过率	0/4 (0%)	64%	提交次数	8	3		
代码效率			代码规范及可读性				
	TA的	参考	代码规范得分				
运行时间	0ms	1s	Line 2: 'CLASS_DEF' should be separated from previous statement. [EmptyLineSeparator]				
占用内存	0K	32768K	Line 6:17: Local variable name 'n' must match pattern '[a-z][a-z0-9][a-zA-Z0-9]*\$'. [LocalVariableName]				
			4.6				

他的代码：

做题用时：99 分钟 语言：Java 运行时间：0ms 占用内存：0K 程序状态：编译错误

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNext()){
            int n = sc.nextInt();
            int option = sc.nextInt();
            List<person> list = new ArrayList<>();
            for (int i = 0; i < n; i++) {
                list.add(new person(sc.next(),sc.nextInt()));
            }
        }
    }
}
```

```
if (option == 0){  
    list.sort((o1, o2) -> o2.core-o1.core);  
}else if (option == 1){  
    list.sort((o1, o2) -> o1.core - o2.core);  
}  
for (person ps : list){  
    System.out.println(ps.name+" "+ps.core);  
}  
}  
}
```



[点此](#)或手机扫描二维码查看代码编写过程