



设顺序循环队列 $Q[0: M-1]$ 的头指针和尾指针分别为 F 和 R . 头指针 F 总是指向队头元素的前一位置. 尾指针 R 总是指向队尾元素的当前位置. 则该循环队列中的元素个数为 ()

正确答案: A 你的答案: 空 (错误)

$(R-F+M) \% M$

$R-F$

$(F-R+M) \% M$

$F-R$

算法的时间复杂度取决于 ()

正确答案: A C 你的答案: 空 (错误)

待处理数据的状态

处理器的速度

问题的规模

程序所占空间

```
1  int f(int x)
2      return ((x>2) ? x*f(x-1) : 3);
3  }
4  int i;
5  I=f(f(2));
```

执行如上函数后, I 的值为 ()

正确答案: C 你的答案: 空 (错误)

30

无限递归

9

2160

有关操作系统常用调度算法叙述正确的是 ()

正确答案: A D 你的答案: 空 (错误)

FCFS 调度算法不利于短作业

SPF 调度算法不利于短作业

HRN 调度算法不利于长作业

HRN 调度算法既利于短作业又有利于长作业

promise 的状态有 ()

正确答案: A C D 你的答案: 空 (错误)

Unfulfilled

Pause





Resolved

Rejected

下面的哪个选项是自底向上分析方法 ()

正确答案: D 你的答案: 空 (错误)

递归下降分析法

预测分析法

LL (1) 分析法

算符优先分析法

有关曼彻斯特编码叙述正确的是 ()

正确答案: C D 你的答案: 空 (错误)

在信号位开始时不改变信号极性, 表示辑“1”

模拟数据转换为数字信号

在信号位中电平从高到低跳变表示 0

数字数据转换为数字信号

关于中间件特点的描述, 不正确的是 ()

正确答案: A 你的答案: 空 (错误)

中间件运行于客户机/服务器的操作系统内核中, 提高内核运行效率

中间件应支持标准的协议和接口

中间件可运行于多种硬件和操作系统平台上

跨越网络, 硬件, 操作系统平台的应用或服务可通过中间件透明交互

下面关于 **promise** 的说法中, 错误的是 ()

正确答案: D 你的答案: 空 (错误)

`resolve ()` 和 `reject` 都是直接生成一个进入相应状态的 `promise` 对象, 其参数就是进入相应状态时传递过去的参数, 可以在完成回调的参数中得到

`Promise.resolve (value)`, `Promise.reject (reason)` 是 `Promise` 构造器上还直接提供了一组静态方法

在调用 `then` 方法或者 `catch` 方法时都是异步进行的, 但是执行速度比较快

`Promise` 构造器的 `prototype` 上还有两个方法, 分别是 `then` 和 `catch`。这两个方法的参数也是回调函数, 这些函数会在 `Promise` 实例进入不同状态后被调用。`then` 对应到 `resolve`, `catch` 对应到 `reject`

KMP 算法下, 长为 n 的字符串中匹配长度为 m 的子串的复杂度为 ()

正确答案: B 你的答案: 空 (错误)

$O (N)$

$O (M+N)$



☐ (M+LOGM)

☐ (N+LOGM)

通过构建有序序列，对于未排序数据，在已排序序列中从后向前扫描，找到相应的位置并插入的排序算法是（）

正确答案: D 你的答案: 空 (错误)

归并排序

选择排序

希尔排序

插入排序

执行 `chmod 644 mm.txt` 后的权限是（）

正确答案: A 你的答案: 空 (错误)

`-rw-r--r--`

`rw-r--r--`

`-rwxr--r--`

`rwxr- -rw-`

下列说法正确的是（）

正确答案: A B D 你的答案: 空 (错误)

二维以上的数组其实是一种特殊的广义表

数组一旦建立，结构的元素个数和元素间的关系就不再变化

数组是一种线性结构，因此只能用来存储线性表

数组采用顺序存储方式表示

为了提高数据库的性能，需要针对系统设计基准测试进行压力测试，那么进行压力而是时需要考虑以下哪些指标（）

正确答案: A B C D 你的答案: 空 (错误)

可扩展性

响应时间

开发性

吞吐量

假如在 MySQL 中有存储过程 `proc1`（员工编号，月份）用来查询员工的工资，两个参数类型均为字符类型，则下列调用存储过程的方法正确的是（）

正确答案: A C 你的答案: 空 (错误)

`call Proc1 ('emp001' , ' 201601');`

`exec Proc1 ('emp001' , ' 201601');`

`call Proc1 ('emp001' , null);`





```
call Proc1 ('emp001' , );
```

利用时间重叠概念实行并行处理的是 ()

正确答案: B 你的答案: 空 (错误)

多处理机

流水处理机

相联处理机

并行 (阵列) 处理机

下列关于策略处理规则描述正确的是 ()

正确答案: A C D 你的答案: 空 (错误)

如果子容器的某个策略被配置, 则此配置值会覆盖由其父容器所传递下来的配置值
当组策略的用户配置和计算机配置冲突的时候, 优先处理用户配置

组策略的配置值是有累加性的

系统是先处理计算机配置, 再处理用户配置

设哈夫曼树中的结点总数为 49, 若用二叉链表作为存储结构, 则该哈夫曼树中总共
有多少个空指针域 ()

正确答案: C 你的答案: 空 (错误)

51

52

50

49

Linux 中包括两种链接: 硬链接 (Hard Link) 和软连接 (Soft Link), 下列说法正
确的是 ()

正确答案: A 你的答案: 空 (错误)

软连接可以跨文件系统进行连接, 硬链接不可以
当删除原文件的时候硬链接文件仍然存在, 且内容不变
硬链接被删除, 磁盘上的数据文件会同时被删除
硬链接会重新建立一个 inode, 软链接不会

在 java 中重写方法应遵循规则的包括 ()

正确答案: B C 你的答案: 空 (错误)

访问修饰符的限制一定要大于被重写方法的访问修饰符
可以有不同的访问修饰符
参数列表必须完全与被重写的方法相同
必须具有不同的参数列表





以下集合对象中哪几个是线程安全的 ()

正确答案: C D 你的答案: 空 (错误)

LinkedList

ArrayList

Vector

Hashtable

下列有关 java 构造跟使用叙述正确的是 ()

正确答案: C D 你的答案: 空 (错误)

构造器的返回值为 void 类型

如果一个源文件中有多个类，那么构造器必须与公共类同名

构造器可以有 0 个，1 个或一个以上的参数

每个类可以有一个以上的构造器

```
1 public class Test{
2     static int cnt = 6;
3     static{
4         cnt += 9;
5     }
6     public static void main(string[] args) {
7         System.out.println ( "cnt =" + cnt);
8     }
9     static{
10         Cnt /=3;
11     };
12 }
```

Cnt 的值是

正确答案: A 你的答案: 空 (错误)

Cnt=5

Cnt=2

Cnt=3

Cnt=6

```
1 b = 2*a/ 4
2 a = "one"
3 print(a,b)
```

执行下列语句后的显示结果是什么 ()

正确答案: C 你的答案: 空 (错误)

10

one0.5





```
error
one 0
```

```
1  1.bit = input("Enter a binary digit:")
2  2.if bit  = 0 or 1:
3  3.      print "your input is",bit
4  4.else
5  5.      print "your input is invalid"
```

以上程序要求用户输入二进制数字 0/1 并显示之，请指出程序中的错误：（）

正确答案: A D 你的答案: 空 (错误)

4
5
3
2

下列有关 **this** 指针使用方法的叙述正确的是（）

正确答案: D 你的答案: 空 (错误)

保证基类保护成员在子类中可以被访问

保证基类私有成员在子类中可以被访问

保证基类共有成员在子类中可以被访问

保证每个对象拥有自己的数据成员，但共享处理这些数据的代码

设有定义 `char *p[]={"Shanghai","Beijing","Honkong"};`则结果为 j 字符的表达式是()

正确答案: B 你的答案: 空 (错误)

```
*p[1] +3
*(p[1] +3)
*(p[3] +1)
p[3] [1]
```

```
1  void func(char *p)
2  {
3      p=p+1;
4  }
5  int main()
6  {
7      char s[]={'1','2','3','4'};
8      func(s);
9      printf("%c",*s);
10     return 0;
11 }
```

以下程序执行后的输出结果为（）





正确答案: C 你的答案: 空 (错误)

2

编译错误

1

无法确定

以下字符串定义与赋值中，正确的是（）

正确答案: D 你的答案: 空 (错误)

```
char s[80] = {"A", "B", "C"}
char s[80]; s = {'A', 'B', 'C'}
char s[80]; s = "ABC";
char s[80] ="ABC";
```

在 MySQL 中，下列关于触发机器的描述正确的是（）

正确答案: A C 你的答案: 空 (错误)

MySQL 的触发器只支持行级出发，不支持语句级触发

触发器可以调用将数据返回客户端的存储程序

在 MySQL 中，使用 new 和 old 引用触发器中发生的记录内容

在触发器中可以使用显示或者隐式方式开始或结束事务的语句

对于工资表结构如下（员工编号，姓名，部门，工资），如果要对查询的结果按照部门升序与工资降序进行排序，则下列排序正确的是（）

正确答案: B C 你的答案: 空 (错误)

```
order by 部门, 工资
order by 部门, 工资 desc
order by 部门 asc, 工资 desc
order by 部门 desc, 工资 desc
```

为了提高数据的查询效率，需要在数据库中建立索引，则下列设计索引的原则描述正确的是（）

正确答案: A B 你的答案: 空 (错误)

在频繁进行排序或分组（即进行 group by 或 order by 操作）的列上建立索引

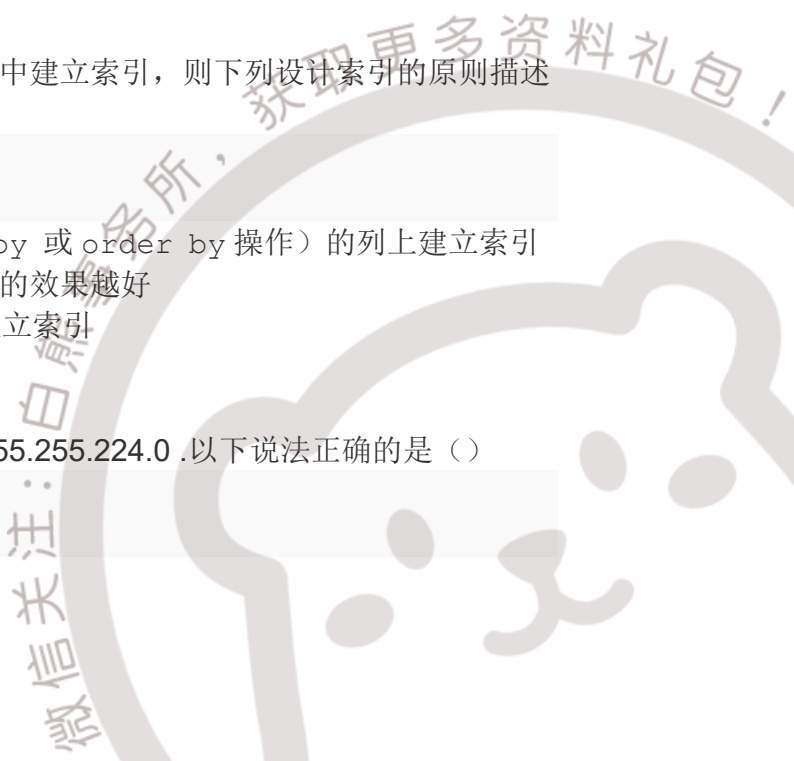
考虑列中值的分布，列的基数越大，索引的效果越好

在 select 关键字后选择列表的列上，建立索引

在表中，索引越多越好

网络地址为 172.16.0.0，采用子网掩码 255.255.224.0 .以下说法正确的是（）

正确答案: D 你的答案: 空 (错误)





其中一个子网的广播地址为: 172.16.32.255
其中一个子网的广播地址为: 172.16.128.255
划分了 8 个有效子网
划分了 6 个有效子网

对下列常见的各种网络术语描述正确的是 ()

正确答案: C D 你的答案: 空 (错误)

ADNS 是一种用于 TCP/IP 应用程序的分布式数据库, 因此它在 TCP/IP 体系中处于应用层

TFTP 是一种文件传递应用程序, 它使用的传输层协议是 TCP

Ping 是对两个 TCP/IP 系统连通性进行测试的基本工具, 它利用 ICMP 进行基本的请求的应答

Telnet 是标准的提供远程登录功能的应用, 可以在不同 OS 系统的主机之间运行

Arthur 最近搬到了新的别墅, 别墅特别大, 原先的桌子显得比较小, 所以他决定换一张新的桌子。他买了一张特别大的桌子, 桌子是由很多条桌腿进行支撑的, 可是回到家之后他发现桌子不稳, 原来是桌子腿长度不太相同。他想要自己把桌子修理好, 所以他决定移除掉一些桌腿来让桌子变得平稳。桌子腿总共有 n 条腿, 第 i 条腿长度为 li , Arthur 移除第 i 桌腿要花费代价为 di 。假设 k 条腿桌子平稳的条件: 超过一半桌腿能够达到桌腿长度的最大值。例如: 一条腿的桌子是平稳的, 两条腿的桌子腿一样长时是平稳的。请你帮 Arthur 计算一下是桌子变平稳的最小总代价。

```
1  大体思路是把先把输入的腿长映射到一个 106 维的腿长数目数组, 数组中的值为对应腿长的
2  没碰到一个腿长, 将比他高的腿长砍掉, 同时计算还需要砍多少桌腿, 对剩余的桌腿排序砍
3  #include<iostream>
4  #include<vector>
5  #include<algorithm>
6  using namespace std;
7
8  int main()
9  {
10         int n;
11         while(cin>>n)
12         {
13                 vector<int >l,d,d_cut;
14                 int a[106]={0};
15                 int li,di;
16                 for(int i=0;i<n;i++)
17                 {
18                         cin>>li;
19                         l.push_back(li);
20                         a[li]++;
21                 }
```





```
22         int minCost=0;
23         for(int i=0;i<n;i++)
24         {
25             cin>>di;
26             d.push_back(di);
27             minCost+=di;
28         }
29         for(int i=105;i>0;i--)
30         {
31             int cost=0;
32             d_cut.clear();
33             if(a[i])
34             {
35                 int cutLegNum=l.size()-a[i];
36                 for(int j=i+1;j<=105;j++)
37                 {
38                     cutLegNum-=a[j];
39                 }
40                 cutLegNum-=a[i]-1;
41                 for(int k=0;k<n;k++)
42                 {
43                     if(l[k]<i)
44                         d_cut.push_back(d[k]);
45                     if(l[k]>i)
46                         cost+=d[k];
47                 }
48                 if(cutLegNum>0)
49                 {
50                     sort(d_cut.begin(),d_cut.end());
51                     for (int k = 0; k <cutLegNum; k++)
52                         cost+=d_cut[k];
53                     if(cost<minCost)
54                         minCost=cost;
55                 }
56             }
57         }
58         cout<<minCost<<endl;
59     }
60     return 0;
61 }
```

现有一个 n 个整数的序列，你要做的就是交换两个数的位置直到整个序列按照升序排列，那么将这个整数序列排好序，需要交换多少次？例如，1, 2, 3, 5, 4，我们只需要交换一次，即将 5 和 4 交换即可。



```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int N = sc.nextInt();

        int[] a = new int[N];

        for (int i = 0; i < N; i++) {

            a[i] = sc.nextInt();

        }

        int count = 0;

        int temp = 0;

        for (int i = 0; i < N - 1; i++) {

            for (int j = 0; j < N - 1 - i; j++) {

                if (a[j + 1] < a[j]) {

                    temp = a[j];

                    a[j] = a[j + 1];

                    a[j + 1] = temp;

                    count++;

                }

            }

        }

        System.out.println(count);

    }

}
```

就是冒泡交换的次数，每次把最大的值放在最后，题目意思应该是每次只能相邻的交换，所以冒泡交换的次数就是最后的结果

