在 Java 中下面哪个对类的声明是错误的?

正确答案: C

```
public class MyClass{}
class MyClass extends MySuperClass implements YourInterface
{}
class MyClass extends MySuperClass1, MySupperClass2 {}
abstract class MyClass implements YourInterface1,
Youriterface2 {}
```

以下哪个选项不是单例模式的优点?

正确答案: D

减少内存开支
减少系统调用
避免资源的多重占用
线程安全

某台计算机连接了 8 个相同的设备，有 N 个进程在竞争使用，每个进程最多会同时占用 3

个设备，请问当 N 大于等于多少时，系统可能发生死锁?

正确答案: C

2
3
4
5

以下哪种操作不会导致计算机从用户态切换至内核态?

正确答案: B　你的答案: 空 (错误)

访问内存时出现缺页异常
对一个变量进行取模运算
创建一个子进程
读取硬盘中文件的内容

TCP 协议在常见的七层网络模型中属于哪一层?

正确答案: A 你的答案: 空 (错误)

传输层
网络层
会话层
数据链路层

UDP 是一种无连接的网络协议，那么一下哪个选项不是 UDP 协议报头的内容?

正确答案: A 你的答案: 空 (错误)

序号（Sequence Number）
源端口号（Source port）和目标端口号（Destination port）
报文长度（Length）
校验和（Checksum）

以下哪个排序算法是稳定的

正确答案: C 你的答案: 空 (错误)

快速排序
选择排序
冒泡排序
堆排序

一颗有 512 个节点的完全二叉树的高度是多少

正确答案: B 你的答案: 空 (错误)

9
10
11
12

- 

- 以下数据库事务的隔离级别中哪一个可能造成脏读

- 正确答案: A 你的答案: 空 (错误)

- 读取未提交内容（Read Uncommited）
- 读取提交内容（Read Committed）
- 可重复读（Repeatable Read）
- 可串行化（Serializable）

如果想列出当前目录以及子目录下所有扩展名为“.txt”的文件，那么可以使用以下哪个命令？

正确答案: A　你的答案: 空 (错误)

```
find . -name "*.txt"
grep ".txt" -r *
ls "*.txt"
less "*.txt"
```

- 

- 以下哪一项不是 c++11 新引入的容器

  - 正确答案: B　你的答案: 空 (错误)

  - std::array
  - std::map
  - std::unordered_map
  - std::forward_list

- 使用 gcc 编译一份 C 代码的过程，报错提示"undefined reference to 'XXXXX'"，这是哪个阶段出错了？

  - 正确答案: D　你的答案: 空 (错误)

  - 预处理
  - 编译
  - 汇编
  - 链接

- 在以下哪种容器上，不能应用二分查找算法？

  - 正确答案: C　你的答案: 空 (错误)

  - std::vector

- `std::deque`
- `std::list`
- `std::array`

- 访问主存上的数据，大概需要多少个机器时钟?

> - 正确答案: C　你的答案: 空 (错误)

- 2
- 10
- 100
- 10000

- 以下哪一项不能有效利用程序的局部性?

> - 正确答案: B　你的答案: 空 (错误)

- 顺序读取数据对象
- 将相关代码拆散到多个 `c` 文件中
- 精简程序 `binary` 的大小
- 将主要的计算逻辑集中在内部循环并做优化

- 以下哪一项不会导致 C 程序发生“段错误”？

> - 正确答案: A　你的答案: 空 (错误)

- 忘记释放已分配的内存块
- 引用不存在的变量
- 引用已经被释放的内存块
- 访问数组越界

- “定义了一系列算法，并将每个算法封装起来，使它们可以相互替换”是指以下哪种设计模式?

> - 正确答案: B　你的答案: 空 (错误)

- 模板模式
- 策略模式
- 状态模式
- 命令模式

- 2019! 的末尾有多少个零?

- 正确答案: B 你的答案: 空 (错误)

- 501
- 502
- 503
- 504

- tcp 连接建立需要几次握手

- 正确答案: C 你的答案: 空 (错误)

- 1
- 2
- 3
- 4

- 以下哪种 TCP 状态需要等待 2MSL

- 正确答案: A 你的答案: 空 (错误)

- TIME_WAIT
- CLOSE_WAIT
- CLOSING
- FIN_WAIT

- 路由器工作在网络协议的哪一层

- 正确答案: C 你的答案: 空 (错误)

- 物理层
- 链路层
- 网络层
- 应用层

- 虚拟内存的容量只受()的限制

  - 正确答案: D 你的答案: 空 (错误)

- 物理内存的大小
- 磁盘空间的大小
- 数据存放的实际地址
- 计算机地址位数

- 以下哪个步骤会产生汇编代码文件

  - 正确答案: B 你的答案: 空 (错误)

- 预处理
- 编译
- 汇编
- 链接

若处理器有 32 位地址，则它的虚拟地址空间为(　)

正确答案: B 你的答案: 空 (错误)

2G
4G
512M
256M

- 按照二叉树的定义，具有 3 个结点的二叉树有几种。

正确答案: C 你的答案: 空 (错误)

3
4
5
6

- 以下哪个不是栈的基本操作

正确答案: A  你的答案: 空 (错误)

删除栈底的元素
删除栈顶元素
判断栈是否为空
栈置空

- 

- 堆排序的时间复杂度为

- 正确答案: B  你的答案: 空 (错误)

- n
- nlogn
- logn
- n^2

- 有 6 个元素 6,5,4,3,2,1 的顺序进栈，问下列哪一个不是合法的出栈序列

- 正确答案: C  你的答案: 空 (错误)

- 5,4,3,6,1,2
- 4,5,3,1,2,6
- 3,4,6,5,2,1
- 2,3,4,1,5,6

原地翻转句子中单词的顺序，但单词内字符的顺序不变。要求：空间复杂度 O(1)，时间复杂度 O(n)。

```
//参考《剑指 offter》第 48 题，翻转字符串
//时间复杂度 0（n），空间复杂度 0（1）

#include<iostream>
(720)#include<string>
using namespace std;
void Reverse(string &s, int left, int right)
{
    if (s.empty())
        return;
    while (left<right)
    {
```

```
        char temp = s[left];
        s[left] = s[right];
        s[right] = temp;
        left++, right--;
    }
}
void ReverseSentence(string &s)
{
    if (s.empty())
        return;

    int left = 0, right = s.size() - 1;

    //反转整个句子
    Reverse(s, left, right);


    //翻转句子中的每个单词
    left = right = 0;
    while (left<s.size())
    {
        if (s[left] == ' ')
        {
            left++;
            right++;
        }
        else if (s[right] == ' ' || right == s.size())
        {
            Reverse(s, left, --right);
            left = ++right;
        }
        else {
            right++;
        }
    }
}
int main()
{
    string s;
    getline(cin, s);
    ReverseSentence(s);
    cout << s;
}
```

输入一个正整数数组，将它们连接起来排成一个数，输出能排出的所有数字中最小的一个。

```
import java.util.*;

public class Test {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String input = sc.next();
        String[] arr = input.split(",");
        Arrays.sort(arr, new Comparator<String>() {
            @Override
            public int compare(String o1, String o2) {
                return (o1 + o2).compareTo(o2 + o1);
            }
        });
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i]);
        }
    }
}
```

有为 N 件物品，它们的重量 w 分别是 w1,w2,...,wn，它们的价值 v 分别是 v1,v2,...,vn，每件物品数量有且仅有一个，现在给你个承重为 M 的背包，求背包里装入的物品具有的价值最大总和?

```
import java.util.*;

public class Main {
    static  int maxValue = 0;
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNext()) {
            int N = sc.nextInt();
            int M = sc.nextInt();
            int[] w = new int[N];
            int[] v = new int[N];
            for (int i = 0; i < N; i++) {
                w[i] = sc.nextInt();
            }
            for (int i = 0; i < N; i++) {
                v[i] = sc.nextInt();
            }
            dfs(w,v,M,0,N,0);
            System.out.println(maxValue);
        }
```

```
    }
    public static void dfs(int[] w, int[] v, int m, int start, int N,int
value) {
        // 剪枝，当 m<=0 不能装了，就不用挨个比较了
        if (m <= 0) {
            return;
        }
        for (int i = start; i < N; i++) {
            // 如果背包容量还够，就装进去
            if (m - w[i] >= 0) {
                value += v[i];
                if (value > maxValue) {
                    maxValue = value;
                }
                m = m - w[i];
                // 在第一个装的基础上，看看后面的装不装
                dfs(w,v,m,i+1,N,value);
                // 回溯，第一个不装，下一轮 for 循环就会考虑装不装第二
个了
                value -= v[i];
                m += w[i];
            }
        }
    }
}
```