1.下面代码的运行结果是（ C ）

```
public class Test{
    public static void main (String[] args){
        List<String> a = null;
        test(a);
        System.out.println(a.size());
    }
    public static void test(List<String> a){
        a=new arrayList<String>();
        a.add(“abc” );
    }
}
```

A . 0
B . 1
C . Java.lang.NullPointerException
D . 以上都不正确

2.Linux 下查看进程占用的 CPU 的百分比， 使用工具（ A ）

A. Ps
B. Cat
C. More
D. Sep

3.JVM 内存里哪个区域不可能发生 OutOfMerncyError(A)

A. 程序计数器
B. 堆
C. 方法区
D. 本地方法栈

4.下面关于阻塞队列（ java.util.concurrent.BlockingQueue ）的说法不正确的是(C)

A. 阻塞队列是线程安全的
B. 阻塞队列的主要应用场景是“生产者-消费者”模型
C. 阻塞队列里的元素不能为 null
D. 阻塞队列的实现必须显示地设置容量

5.如果现在需要创建一组任务， 他们并行的执行工作，然后进行下一个步骤之前等待，直至所有的任务都完成，而去这种控制可以重用多次， 这种情形使用 java.util.concurrent包中引入哪种同步工具最适合（ B）

A. CountDownLatch
B. CyclicBarrier
C. Semaphore
D. FutureTask

6.java 中， 为什么基类不能做为 HashMap 的键值， 而只能是引用类型，把引用类型作为 HashMap 的键值， 需要注意哪些地方？

答案：引用类型和原始类型的行为完全不同，并且它们具有不同的语义。引用类型和原始类型具有不同的特征和
用法，它们包括：大小和速度问题，这种类型以哪种类型的数据结构存储，当引用类型和原始类型用作某个类的实例数据时所指定的缺省值。对象引用实例变量的缺省值为 null，而原始类型实例变量的缺省值与它们的类型有关。

7.编写一个工具类 StringUtil, 提供方法 int compare(char[] v1 ,char[] v2)方法，比较字符串v1,v2 ,如果按照字符顺序 v1>v2 则 return 1 ,v1=v2 则 return 0, v1<v2 则 return-1。

```
public class StringUtil{
    int compare(char[] v1,char[] v2) {
        String str1 = new String(v1);
        String str2 = new String(v2);
        int result = str1.compareTo(str2);
        return result == 0 ? 0 : (result > 0 ? 1 : -1);
}
```

8.Java 出现 OutOfMemoryError(OOM)的原因有那些？出现 OOM 错误后，怎么解决？

触发 java.lang.OutOfMemoryError:最常见的原因就是应用程序需要的堆空间是大的，但是 JVM 提供的却小。
这个的解决方法就是提供大的堆空间即可。除此之外还有复杂的原因：

内存泄露：特定的编程错误会导致你的应用程序不停的消耗更多的内存，每次使用有内存泄漏风险的功能就会留
下一些不能被回收的对象到堆空间中，随着时间的推移，泄漏的对象会消耗所有的堆空间，最终触发java.lang.OutOfMemoryError: Java heap space 错误。

解决方案：
第一个解决方案是显而易见的，你应该确保有足够的堆空间来正常运行你的应用程序，在 JVM 的启动配置中增加
如下配置：-Xmx1024m

流量/数据量峰值：应用程序在设计之初均有用户量和数据量的限制，某一时刻，当用户数量或数据量突然达到一
个峰值，并且这个峰值已经超过了设计之初预期的阈值，那么以前正常的功能将会停止，并触发 java.lang.OutOfMemoryError: Java heap space 异常

解决方案，如果你的应用程序确实内存不足，增加堆内存会解决 GC overhead limit 问题，就如下面这样，给你的应用程序 1G 的堆内存：

```
java -Xmx1024m com.yourcompany.YourClass
```