1、在软件开发过程中，我们可以采用不同的过程模型，下列有关 增量模型描述正确的是（）

是一种线性开发模型，具有不可回溯性

把待开发的软件系统模块化，将每个模块作为一个增量组件，从而分批次地分析、设计、编码和测试这些增量组件

适用于已有产品或产品原型（样品），只需客户化的工程项目

软件开发过程每迭代一次，软件开发又前进一个层次

解析：增量模型是把待开发的软件系统模块化，将每个模块作为一个增量组件，从而分批次地分析、设计、编码和测试这些增量组件。运用增量模型的软件开发过程是递增式的过程。相对于瀑布模型而言，采用增量模型进行开发，开发人员不需要一次性地把整个软件产品提交给用户，而是可以分批次进行提交。

2、下面有关值类型和引用类型描述正确的是（）？

值类型的变量赋值只是进行数据复制，创建一个同值的新对象，而引用类型变量赋值，仅仅是把对象的引用的指针赋值给变量，使它们共用一个内存地址。

值类型数据是在栈上分配内存空间，它的变量直接包含变量的实例，使用效率相对较高。而引用类型数据是分配在堆上，引用类型的变量通常包含一个指向实例的指针，变量通过指针来引用实例。

引用类型一般都具有继承性，但是值类型一般都是封装的，因此值类型不能作为其他任何类型的基类。

值类型变量的作用域主要是在栈上分配内存空间内，而引用类型变量作用域主要在分配的堆上。

解析：B错在，值类型变量不包含实例，实例是针对于对象的概念，当类实例化为对象的时候，这个时候可以称为是类的一个实例。同时，效率比较高这个概念比较模糊。C错在，封装的概念也是针对类而言的，值类型数据不存在封装概念。D错在，值类型变量可以作为成员变量存储在堆里，例如一个Class A中包含一个int value，那么value是作为成员变量存储在堆中的。D选项表述有漏洞。

3、如何在多线程中避免发生死锁？

允许进程同时访问某些资源。

允许进程强行从占有者那里夺取某些资源。

进程在运行前一次性地向系统申请它所需要的全部资源。

把资源事先分类编号，按号分配，使进程在申请，占用资源时不会形成环路。

解析：死锁：在多道程序设计环境下，多个进程可能竞争一定数量的资源，。一个进程申请资源，如果资源不可用，那么进程进入等待状态。如果所申请的资源被其他等待进程占有，那么该等待的进程有可能无法改变状态，这种情况下称之为死锁。

死锁的四个条件：互斥：至少有一个资源必须处在非共享模式，即一次只能有一个进程使用，如果另一进程申请该资源，那么申请进程必须延迟直到该资源释放为止。占有并等待：一个进程必须占有至少一个资源，并等待另一个资源，而该资源为其他进程所占有。非抢占：资源不能被抢占。循环等待：有一组进程{P0,P1,...Pn},P0等待的资源被P1占有，P1等待的资源被P2占有，Pn-1等待的资源被Pn占有，Pn等待的资源被P0占有。

4、以下为求0到1000以内所有奇数和的算法，从中选出描述正确的算法（ ）

①s=0；②i=1；③s=s+i；④i=i+2；⑤如果i≤1000，则返回③；⑥结束

①s=0；②i=1；③i=i+2；④s=s+i；⑤如果i≤1000，则返回③；⑥结束

①s=1；②i=1；③s=s+i；④i=i+2；⑤如果i≤1000，则返回③；⑥结束

①s=1；②i=1；③i=i+2；④s=s+i；⑤如果i≤1000，则返回③；⑥结束

解析：D算的是1+3+。。。+10001，A是0+1+3+。。。+999,就是D多算了一个

5、关于递归法的说法不正确的是（ ）

程序结构更简洁

占用CPU的处理时间更多

要消耗大量的内存空间，程序执行慢，甚至无法执行

递归法比递推法的执行效率更高

解析：递归就是一层一层的调用函数进栈，会大量存储重复的数据，占用大量内存，且会有栈溢出的可能。而递推就是不用存储重复的数据，从底层获得了数据直接递给上一层即可。

6、字符串”ABCD”和字符串”DCBA”进行比较，如果让比较的结果为真，应选用关系运算符（）

> < = >=

解析：比较的时候比的是ASCII码，从第一位开始比，如果第一位能比较出谁大谁小则直接返回，若相等，比较下一个字符，因此选B

7、以下运算符中运算优先级最高的是（ ）

+ OR > \

解析：

以上是我整理的运算符优先级，数字越小表示优先级越高。在这里我选的是+号，如果+号表示的是正号的话那么优先级是大于除/号的，这题目略微的不严谨，望周知。

8、采用哪种遍历方法可唯一确定一棵二叉树？（  ）

给定一棵二叉树的先序和后序遍历序列

给定一棵二叉树的后序和中序遍历序列

给定先序、中序和后序遍历序列中的任意一个即可

给定一棵二叉树的先序和中序遍历序列

解析：前序加中序  或  后序加中序都是可以的。

9、已知小顶堆：{51,32,73,23,42,62,99,14,24,3943,58,65,80,120}，请问62对应节点的左子节点是（没有正确答案）

99 73 3943 120

解析：

10、若串S=”UP！UP！JD”，则其子串的数目

33 37 39 35

解析：一个八个字符串 公式n(n+1)/2 + 1

11、一颗二叉树的叶子节点有5个，出度为1的结点有3个，该二叉树的结点总个数是？

11 12 13 14

解析：因为是二叉树，所以度不能大于2。【度（子节点个数）】。5个叶子节点，三个度为1，则有上一层有4个，叶子节点数=度为2的节点个数+1=5；所以度为2的节点数为4；总节点数 = 度为1的节点数 + 度为2的节点数 + 叶子节点数 = 3 + 4 +5 = 12。

12、权值分别为9、3、2、8的结点，构造一棵哈夫曼树，该树的带权路径长度是？

36 40 45 46

解析：计算带权路径长度：结点的带权路径长度为从根结点到该结点之间的路径长度与该结点的权的乘积=3\*2+3\*3+2\*8+1\*9=40

13、在（）中，只要指出表中任何一个结点的位置，就可以从它出发依次访问到表中其他所有结点。

线性单链表

双向链表

线性链表

循环链表

解析：循环链表只要沿一个方向一直走下去就可以遍历。

14、打电话使用的数据传输方式是（），手机上网使用的数据传输方式是（）？

电路交换，电路交换

电路交换，分组交换

分组交换，分组交换

分组交换，电路交换

解析：电路交换有预留，且分配一定空间，提供专用的网络资源，提供有保证的服务，应用于电话网；分组交换无预留，且不分配空间，存在网络资源争用，提供有无保证的服务。分组交换可用于数据报网络和虚电路网络。我们常用的Internet就是数据报网络，单位是Bit。

15、后退N帧协议的发送窗口大小是（），接收窗口的大小是（）？

=1,=1

=1,>1

>1,>1

>1,=1

解析：滑动窗口协议有：1、停止等待协议，发送窗口=1，接受窗口=1；2、退后N帧协议，发送>1,接收=1;3、选择重传协议，发送>1,接收>1;

16、网络管理员把优盘上的源代码给程序员参考，但要防止程序误删除或修改，以下正确的加载方式是（      ）

mount -o defaults /dev/sdb1 /tools

mount -r /dev/sdb1 /tools

mount -o ro /dev/sdb1 /tools

mount -o ro /dev/sdb /tools

解析：

-r: 将文件系统作为只读文件系统进行安装，而不考虑它先前在 /etc/file systems 文件中指定的内容或者先前的任何命令行选项。

ro : 将已安装的文件指定为只读文件，而不考虑它先前在 /etc/file systems 文件中指定的选项或者先前的任何命令行选项。缺省值是 rw。

17、重复的数据，会增加磁盘空间的占有率，延长操作数据的时间。可以使用规范化处理数据冗余，以下对符合第一范式的表述正确的是：

非键属性和键（主键）属性间没有传递依赖

非键属性和键（主键）属性间没有部分依赖

表中不应该有重复组。列重复拆成另外一张表；行重复拆成多行

一个表中的列值与其他表中的主键匹配

解析：指数据库表的每一列(即每个属性)都是不可分割的基本数据项，同一列中不能有多个值，即实体中的某个属性不能有多个值或者不能有重复的属性。简而言之，第一范式就是无重复的列。

18、以下哪条SQL语句可以返回table1中的全部的key：

select tabel1.key from table1 join tabel2 on table1.key=table2.key

select tabel1.key from table1 right outer join tabel2 on table1.key=table2.key

select tabel1.key from table1 left semi join tabel2 on table1.key=table2.key

select tabel1.key from table1 left outer join tabel2 on table1.key=table2.key

解析：由于需要返回table1中的全部的key，则需要左连接，而left semi join表示只打印出左边表中 的key，但前提是左表中的key在右表中存在，否则会过滤掉

19、Mysql中表user的建表语句如下，

CREATE TABLE `user` (

  `id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT '主键Id',

  `name` varchar(255) DEFAULT NULL COMMENT '名称',

  `age` int(11) DEFAULT NULL COMMENT '年龄',

  `address` varchar(255) DEFAULT NULL COMMENT '地址',

  `created\_time` datetime DEFAULT NULL COMMENT '创建时间',

  `updated\_time` datetime DEFAULT NULL COMMENT '更新时间',

  PRIMARY KEY (`id`),

  KEY `idx\_com1` (`name`,`age`,`address`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户表';

以下哪个查询语句没有使用到索引idx\_com1？

select \* from user where name='张三' and age = 25 and address='北京大兴区';

select \* from user where name='张三' and address='北京大兴区';

select \* from user where age = 25 and address='北京大兴区';

select \* from user where address='北京大兴区' and age = 25 and name='张三'

解析：索引的最左匹配特性：当b+树的数据项是复合的数据结构，比如(name,age,sex)的时候，b+数是按照从左到右的顺序来建立搜索树的，比如当(张三,20,F)这样的数据来检索的时候，b+树会优先比较name来确定下一步的所搜方向，如果name相同再依次比较age和sex，最后得到检索的数据；但当(20,F)这样的没有name的数据来的时候，b+树就不知道下一步该查哪个节点

20、public class Main {

    private static int x = 10;

    private static Integer y = 10;

    public static void updateX(int value) {

        value = 3 \* value;

    }

    public static void updateY(Integer value) {

        value = 3 \* value;

    }

    public static void main(String[] args) {

        updateX(x);

        updateY(y);

    }

}

执行以上程序后，x和y的值分别是多少？

10,10

10,30

30,10

30,30

解析：形参名value和字段x，字段y不同，运算改变了内存中value，但是xy没有改变。当方法updateX的形参名改为x，此时运算会改变内存中x的值，是30，y同理。总结：仅当形参名和字段名一样，字段值才变。（本题一个叫value，一个叫x/y，不一样，所以不变）

21、JAVA的类加载期负责整个生命周期内的class的初始化和加载工作，就虚拟机的规范来说，以下代码会输出什么结果？

public class Test {

    public static void main(String[] args) {

        System.out.println(Test2.a);

    }

}

class Test2{

    public static final String a="JD";

    static {

        System.out.print("OK");

    }

}

只有JD

只有OK

输出 JDOK

输出 OKJD

解析：

22、继承是JAVA语言的一个特性，针对类的继承，虚拟机会如何进行父类和子类的初始化加载呢？请阅读代码选择出该段代码的输入结果。

public class Test {

    public static void main(String[] args) {

        System.out.print(B.c);

    }

}

class A {

    public static String c = "C";

    static {

        System.out.print("A");

    }

}

class B extends A{

    static {

        System.out.print("B");

    }

}

AC

ABC

C

BC

解析：类加载的初始化阶段会执行静态块的代码，通过子类去调用父类的静态变量子类不会被初始化。

23、根据类加载器加载类的初始化原理，推断以下代码的输入结果为？

public class Test {

    public static void main(String[] args) throws Exception{

      ClassLoader classLoader=ClassLoader.getSystemClassLoader();

      Class clazz=classLoader.loadClass("A");

      System.out.print("Test");

      clazz.forName("A");

    }

}

class A{

    static {

        System.out.print("A");

    }

}

TestA

ATestA

ATest

Test

解析：用ClassLoader加载类，是不会导致类的初始化（也就是说不会执行<clinit>方法）.Class.forName(...)加载类，不但会将类加载，还会执行会执行类的初始化方法。

24、public class Main {

    public static void main(String[] args) {

        System.out.print(fun1());

    }

    public static String fun1() {

        try {

            System.out.print("A");

            return fun2();

        } finally {

            System.out.print("B");

        }

    }

    public static String fun2() {

        System.out.print("C");

        return "D";

    }

}

执行以上程序后，输出结果正确的是？

ABCD

ACDB

ACBD

不确定

解析：try中执行完return的语句后，不返回，执行finally块，finally块执行结束后，返回到try块中，返回try块中最后return的值

25、import java.util.ArrayList;

import java.util.List;

public class Main {

    public static void main(String[] args) {

        List<String> list = new ArrayList<>();

        for(int i=0;i<100;i++){

            list.add("a");

        }

    }

}

JDK1.8中，执行以上程序后，该list进行了几次扩容？

4

5

6

7

解析：初始10，每次扩容为原先的1.5倍

26、import java.util.concurrent.ArrayBlockingQueue;

import java.util.concurrent.ThreadPoolExecutor;

import java.util.concurrent.TimeUnit;

public class Main {

    public static void main(String[] args) {

        ThreadPoolExecutor executor = new ThreadPoolExecutor(5, 10, 15, TimeUnit.SECONDS,

                new ArrayBlockingQueue<Runnable>(5), new ThreadPoolExecutor.CallerRunsPolicy());

    }

}

线程池executor在空闲状态下的线程个数是？

0

5

10

不确定

解析：前三个参数：核心线程有5个，最大线程数是10个，keepAliveTime是15s，如果线程池中的线程大于5，那么超15s的空闲线程就会被结束，也就是说，一定会保持5个线程不会被结束。当所有任务完成后，会保持5个空闲的线程

27、public class Main{

    public static ArrayList<String> list = new ArrayList<>();

    public static void main(String[] args) throws Exception{

        Thread t1 = new Main().new MyThread();

        Thread E10:E64t2 = new Thread(new Main().new MyRunnable());

        t1.setPriority(3);

        t2.setPriority(8);

        t1.start();

        t2.start();

        t2.join();

        for (int i = 0; i < 100000; i++) {

            i++;

        }

        list.add("main");

        t1.join();

        for (String s : list) {

            System.out.println(s);

        }

    }

    class MyThread extends Thread{

        @Override

        public void run(){

            for (int i = 0; i < 100000; i++) {

                i++;

            }

            list.add("Thread 1");

        }

    }

    class MyRunnable implements Runnable{

        @Override

        public void run(){

            for (int i = 0; i < 100000; i++) {

                Thread.yield();

                i++;

            }

            list.add("Thread 2");

        }

    }

}

Thread 1 Thread 2 main

Thread 2 Thread 1 main

Thread 2 main Thread 1

无法判断

解析：Java线程中的Thread.yield( )方法，译为线程让步。顾名思义，就是说当一个线程使用了这个方法之后，它就会把自己CPU执行的时间让掉，

让自己或者其它的线程运行，注意是让自己或者其他线程运行，并不是单纯的让给其他线程。

        yield()的作用是让步。它能让当前线程由“运行状态”进入到“就绪状态”，从而让其它具有相同优先级的等待线程获取执行权；但是，并不能保

证在当前线程调用yield()之后，其它具有相同优先级的线程就一定能获得执行权；也有可能是当前线程又进入到“运行状态”继续运行！

      举个例子：一帮朋友在排队上公交车，轮到Yield的时候，他突然说：我不想先上去了，咱们大家来竞赛上公交车。然后所有人就一块冲向公交车，

有可能是其他人先上车了，也有可能是Yield先上车了。

     但是线程是有优先级的，优先级越高的人，就一定能第一个上车吗？这是不一定的，优先级高的人仅仅只是第一个上车的概率大了一点而已，

最终第一个上车的，也有可能是优先级最低的人。并且所谓的优先级执行，是在大量执行次数中才能体现出来的。

28、系统管理员编写扫描临时文件的shell程序tmpsc.sh, 测试该程序时提示拒绝执行，解决的方法有（      ）

chmod 644 tmpsc.sh

chmod 755 tmpsc.sh

chmod a+x tmpsc.sh

chmod u+x tmpsc.sh

29、在bash编程中,算术比较大于、大于等于的运算符是（      ）

> >= ge gt

解析：EQ 就是 EQUAL等于 、NE就是 NOT EQUAL不等于 、GT 就是 GREATER THAN大于 、LT 就是 LESS THAN小于 、GE 就是 GREATER THAN OR EQUAL 大于等于 、LE 就是 LESS THAN OR EQUAL 小于等于

30、如何获取上一条命令执行的返回码

$! 0 $? $#

解析：$0：Shell本身的文件名。$!：Shell最后运行的后台Process的PID(后台运行的最后一个进程的进程ID号)。$?：最后运行的命令的结束代码（返回值）即执行上一个指令的返回值 (显示最后命令的退出状态。0表示没有错误，其他任何值表明有错误)。$#：添加到Shell的参数个数。

31、Shell 脚本（shell script），是一种为 shell 编写的脚本程序。现有一个test.sh文件，且有可执行权限，文件中内容为：

#!/bin/bash

aa='Hello World !'

请问下面选项中哪个能正常显示Hello World !

sh test.sh >/dev/null 1 && echo $aa

./test.sh >/dev/null 1 && echo $aa

bash test.sh >/dev/null 1 && echo $aa

. ./test.sh >/dev/null 1 && echo $aa

解析：点斜杠，点空格斜杠，两者都可以执行脚本，但是：第一种方式是在子进程中运行脚本，第二种方式是在当前进程中执行脚本。非当前进程是看不到aa变量的

————————————————