1.主机甲与主机乙之间已建立一个TCP连接，主机甲向主机乙发送了两个连续的TCP段，分别包含300B和500B的有效载荷，第一个段的序列号为200，主机乙正确接收到这两个数据段后，发送给主机甲的确认序列号是

200

500

800

1000

2.在支持多线程的系统中，进程P创建的若干个线程不能共享的是

进程P的代码段

进程P中打开的文件

进程P的全局变量

进程P中某线程的栈指针

3.排序算法的效率取决于元素的比较次数与元素的位置移动次数，现需要对数组进行升序排序，已知一数组的元素为{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}，问下面哪种排序算法的效率最高？

插入排序

选择排序

快速排序

冒泡排序

4.一个栈的入栈序列是a b c d e，则栈的输出序列不可能是

dceab

decba

edcba

abcde

5.[多选]对数据库，关于索引的理解正确的是

创建索引能提高数据插入的性能

索引应该根据具体的检索需求来创建，在选择性好的列上创建索引

索引并非越多越好

建立索引可加速查询

6.[多选]用浏览器访问一个Internet网站，可能使用到的协议有

PPP

HTTP

POP

ARP

7.[多选]查找或删除性能较低的数据结构有

有序数组

有序链表

AVL树

Hash表

8. [多选]以下哪些与编译器的任务有关？

公共子表达式合并

运行程序前加载其依赖的动态库

尾递归优化

常量、不变式预计算

9. 现有代码如下，则 func(5)的返回值为 \_\_\_\_\_\_

int func(int n){

if(n <= 1){

return 1;

}else{

return n \* func(n-1);

}

}

10. 下面C程序的运行打印结果是 \_\_\_\_\_

#include <stdio.h>

int main(int argc, char\*\* argv) {

char\* array[] = {"hello", "my", "world", "goodbye"};

char\*\* p = array;

p = p + 2;

printf("%s", \*p);

return 0;

}

11. 一个长度为100的循环链表，指针A和指针B都指向了链表中的同一个节点，A以步长为1向前移动，B以步长为3向前移动，最少需要同时移动 \_\_\_\_\_\_ 步A和B才能再次指向同一个节点

12. 一棵完全二叉树中有33个结点，则该完全二叉树的深度为\_\_\_\_\_

13. {0, 2, 1, 4, 3, 9, 5, 8, 6, 7}是以数组形式存储的最小堆，删除堆顶元素0后的堆的新结果是\_\_\_\_\_（结果需要英文逗号分隔）

14.已知关键字序列为(51,22,83,46,75,18,68,30),进行快速排序，第一趟按关键码字51进行,完成后的序列为\_\_\_\_\_（结果需要英文逗号分隔）

15. 如果下列的公式成立：77+77=121，则数字是采用\_\_\_\_ 进制表示的

16. LRU的cache长度为3,初始为空。依次访问如下元素后，cache里的内容是\_\_\_\_\_\_（结果需要英文逗号分隔）

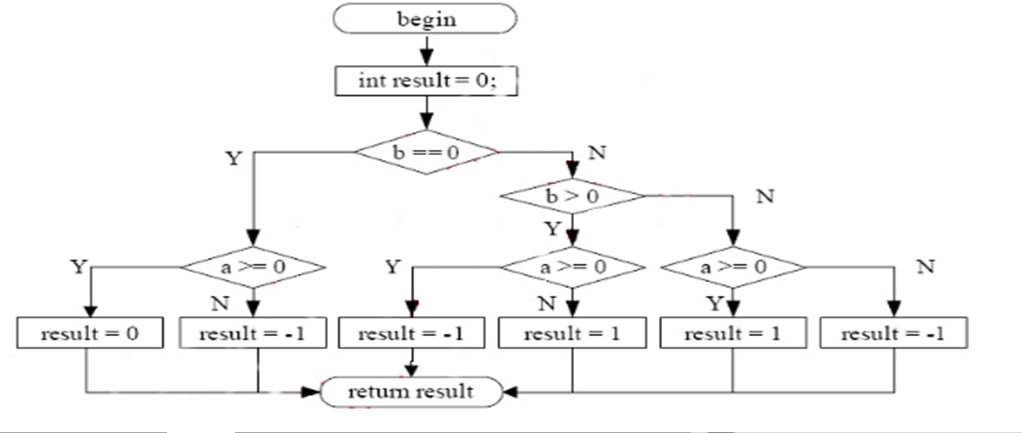
A,A,B,C,A,D,C,E

17. 已知一算数表达式的中缀表达式为 a-(b+c/d)\*e，其后缀形式为\_\_\_\_\_

18. 从1 - 1001中，能被数字2或者数字3或者数字5整除的数字有 \_\_\_\_\_ 个

19. 5个盒子每个里面各有一个球，把球全拿出来打乱再放回去，每个球都不在自己原来的盒子里，有\_\_\_\_\_ 种可能

20. 当用分支覆盖法对以下流程图进行测试时，至少需要设计 \_\_\_\_\_\_个测试用例



21. 题目描述

对一个整数的四则运算后缀表达式，请写函数将其打印成日常我们使用的中缀表达式。如对ab+c\*，打印出 (a+b)\*c 。后缀表达式以一个列表形式作为函数输入，列表的元素为数字或加减乘除操作符。

22. 题目描述

挖雷游戏是一个N\*N格子棋盘，一些随机的格子里有雷，把所有不是雷的格子挖开游戏取胜结束，挖中了任一有雷的格子游戏失败结束。在挖开一个没有雷的格子时，格子上会显示数字，表示相邻的8个格子里有几颗雷，如果是0颗，则程序会帮助把相邻的格子自动全挖开，如果其中又有0颗的，则继续下去。

请你写函数实现对挖开一个0颗雷的格子后，程序自动处理的过程。布了雷的所有格子的坐标作为已知的输入条件。

23. 题目描述

考虑一个网络服务，希望具备防刷的安全特性。假设要求策略是对每次请求访问，如果该请求的来源IP，在当前的前N秒内已经请求过了M次，则拒绝服务X秒。请设计方案，无需写出完全代码，描述清楚设计实现即可。并请针对设计出的方案分析利弊

24. 题目描述

【测试方向优先】你用浏览器打开一个网站，却没有按预期看到应有的网页内容。请分析各种可能的原因，如果这些原因表现出来的现象不同，也请描述

25. 题目描述

【测试方向优先】一个智能玩具，有N个不同模块，每个模块已独立测试过。每个模块有输入输出两个接口，可分别对接任意另外两个模块，（接一个的输出和另一个的输入），这样整个N个模块便可线性组装出多种不同模型（头尾不连成环形）。对任一种模型结果，有现成的通用测试过程。现在为了全覆盖测试，请你写函数生成出所有的模型作为测试用例。（每个模型都必须是用上全部N个模块来组装）

26. 题目描述

【运维方向优先】关系数据库的设计，在内存与硬盘速度不匹配的情况下，为了提高查询速度普遍采用了B树或B+树的存储结构。a. 请解释一下B树或B+树。b. 请描述一个例子过程，说明其相对其他结构（如二叉树）提高查询速度的道理。

27. 题目描述

【运维方向优先】你用微信（或QQ）app给中意的她（他）发送了一句表白，很快收到了一句回复“呵呵”。从你输入完消息点下“发送”按钮，到“呵呵”呈现出来的这段时间，你的手机系统里发生了哪些事情？请根据你所学的计算机知识，尽可能详细的解释。（提示：从软硬件的尽量多的层次考虑和描述。）

答案：

1. D

2. D

3. A

4. A

5. BCD

6. ABD

7. AB

8. ACD

9. 120

10. world

11. 50

12. 6

13. {1,2,5,4,3,9,7,8,6} 或 1,2,5,4,3,9,7,8,6

14. 30,22,18,46,51,75,68,83 或 （30,22,18,46,51,75,68,83）

15. 13

16. E,C,D

17.abcd/+e\*-

18.734

19.44

20.6

21.参考要点：使用一个栈，扫描输入列表入栈，碰到操作符时，将前两个元素出栈，与操作符组成中缀子表达式的字符串再入栈。组成字符串时，要考虑操作符两边的元素是否要加括号。扫描完后栈中应剩下唯一字符串，打印即可。注意输入表达式的错误处理。

示意伪代码：

function printInfixExpression(list postfixExpression)

{ Stack stack;

postfixExpression.foreach( x=> {

if(x is Number) { stack.push(x); }

else if(x is Operator) {

if(stack.size() < 2) { printErrorThenReturn(); }

left = stack.pop(); right = stack.pop();

if (left is ExpressionString and left.operator.priority < x.priority) {

left.text = ‘(‘ + left.text + ‘)’;

}

if (right is ExpressionString and right.operator.priority <= x.priority) { //这里操作符优先级判断要用小于等于

right.text = ‘(‘ + right.text + ‘)’; }

}

subExpr = new ExpressionString;

subExpr.operator = x;

subExpr.text = stringAppend(left, x, right);

stack.push(subExpr);

}

// 既不是数字也不是操作符，输入错误

else { printErrorThenReturn(); }

} // end of lambda

); // end of foreach

if(stack.size() > 1){ // 输入表达式有错误

printErrorThenReturn();

}

print(stack.pop()O); // 栈中剩下唯一元素，为结果表达式字符串。

}

22.参考要点：自动挖开的所有0颗雷的格子是一个连通图（每个节点最多4条边），函数通过计算当前节点周围的雷数量来不断发现相邻节点，边发现边实现图的遍历，遍历过程就是将相应格子的状态从“未挖开”变为“挖开“。

代码容易出现的错误：

遍历时未考虑回到原点形成无限循环的状况，遍历时未考虑雷区N\*N的边界情况

23.

参考要点：

每个IP关联一个最近访问时间戳T，和一个链表L，链表元素为一个细粒度时间段内的访问次数，比如粒度定为500毫秒，则链表内元素依次代表了该IP当前500毫秒的（开始时间，结束时间，请求次数），往前的500毫秒中（开始时间，结束时间，请求次数），再往前500毫秒的（开始时间，结束时间，请求次数）。。。相邻元素的时间段不一定需要连续。

用hash关联IP保存所有T和L

对每一个来访请求，按IP从hash中找到关联的链表L，将最新的元素计数加1（根据当前时间决定是否创建新元素），将N秒前的元素删除，如果剩余元素的计数之和大于等于M，并且最近访问时间戳T到当前未超过X秒，则拒绝服务，否则更新T并提供服务。

定时器将过期IP数据从hash中删除

处理过程中使用同步结构，保证并发安全。

24.参考要点：

人的原因：弄错网址；拼写错误；未打开网络；。。。

本机原因：浏览器版本低不兼容网页；本机防火墙阻止访问；中病毒导致浏览器工作不正常；。。。

服务端原因：服务器宕机；服务器出错；服务器高负载无法及时回应；服务器超负载主动拒绝回应；本机被服务端加入了黑名单；

网络原因：网络不通；网络拥塞；DNS解析失败；DNS解析到错误的IP；。。。

其他环境原因：域名被劫持；访问被黑客攻击；。。。

25.参考要点：实质是写一个函数，生成打印N个不同元素（模块）的所有N!个全排列（模型）。

示意伪代码：递归函数，两个参数，一个是排列前缀，一个是子排列的所有元素，元素使用整数类型。

// 初始调用传入left为空，childrenToPermute为N个元素的list。

function printPermutation(List<int> left, List<int> childrenToPermute)

{

if(childrenToPermute.size() == 0){

printList(left); // 将left中元素依次打印，即为一个排列，也即模型。

return; //递归结束

}

childrenToPermute.foreach( x => {

List<int> newLeft = left.copy().add(x);

List<int> newChildren = childrenToPermute.copy().remove(x);

printPermutation(newLeft, newChildren); // 递归调用

}

);

}

26.【运维方向优先】关系数据库的设计，在内存与硬盘速度不匹配的情况下，为了提高查询速度普遍采用了B树或B+树的存储结构。a. 请解释一下B树或B+树。b. 请描述一个例子过程，说明其相对其他结构（如二叉树）提高查询速度的道理

多叉平衡树；叶节点全部在同一层；非叶节点有n-1个排序关键字和n个子树指针；根节点至少有两个子树，内部节点至少有m/2棵子树；相比B树，B+树内部节点除了关键字和子树指针，不保存数据信息，叶节点包含相应所有关键字和对应数据信息；

例子能说明核心思想即可，要点：1.每读取一个树节点，相当于读一次新的磁盘块，消耗一次磁盘IO；2.同样多的数据量下，B/B+树的高度大大降低，从而查找到数据所在的树节点做的磁盘IO次数少很多；3.树节点读取后，在内存中继续查找出所需数据，要消耗的时间量级与磁盘IO耗时相比小许多。

27.

不考虑电脑OS与手机OS某些细节略存差异。

参考要点：回答是否从硬件（键盘网卡）、OS、TCP协议栈、运行库、浏览器、HTTP、HTML/JS等多层面描述。

参考回答：键盘硬件中断；OS处理中断，转换为特定消息放入浏览器程序的事件队列；浏览器的消息循环处理该消息，请求网址；OS请求本地域名缓存或域名服务器解析网址中的域名，得到IP；浏览器向该IP建立TCP连接（默认80端口）；浏览器发送GET请求，包含网站的路径，TCP协议栈组装为TCP包，通过网卡发送；浏览器等待网站回复，进程被OS切换为等待状态；网站返回的数据到来，网卡产生中断；OS处理中断，TCP协议栈将数据读入buffer；浏览器获得数据，处理HTTP头，显示HTML网页

更多：OS发ARP包获得网关MAC地址，所有DNS请求、网站TCP等数据包均发向该网关；浏览器IO等待期间OS切换运行系统中其他进程；浏览器将HTTP头中解析出的cookie保存到文件系统；根据网页内容发起更多的HTTP请求获取图片、运行内嵌的javascript脚本等；将网页按照HTTP 头的指示缓存；将网址加入浏览历史保存到文件系统；浏览器整个处理过程中，运行库和OS对内存做相应分配释放，磁盘做相应的IO操作；。。。