嵌入式面笔试 题库

日期：2022.04.22

《嵌入式面笔试题库》

提示：

1、 题目类型包括：单选题、多选题、简答题、填空题、编程题等

2、 题目前缀【类型】【知识点】

知识点目录			
嵌入式产品规划	技术评估与设计	产品体验与系统整合	嵌入式硬件基础与实践
Linux	体系与内核基础	接口与应用	感知技术
交互技术	智能应用	嵌入式系统	操作系统
数据结构与算法	计算机组成原理	中间件	云服务设计
混合运服务	语言基础	语言概念	技术评估与设计

面试/笔试题

1.【简答题】【嵌入式硬件基础与实践】同步电路和异步电路的区别是什么？

同步电路：存储电路中所有触发器的时钟输入端都接同一个时钟脉冲源，因而所有触发器的状态的变化都与所加的时钟脉冲信号同步。

异步电路：电路没有统一的时钟，有些触发器的时钟输入端与时钟脉冲源相连，这有这些触发器的状态变化与时钟脉冲同步，而其他的触发器的状态变化不与时钟脉冲同步。

2.【简答题】【嵌入式硬件基础与实践】什么是竞争与冒险现象？怎样判断？如何消除？

在组合逻辑中，由于门的输入信号通路中经过了不同的延时，导致到达该门的时间不一致叫竞争。

产生毛刺叫冒险。判断方法：代数法、图形法（是否有相切的卡诺圈）、表格法（真值表）。

如果布尔式中有相反的信号则可能产生竞争和冒险现象。

冒险分为偏“1”冒险和偏“0”冒险。解决方法：一是添加布尔式的消去项；二是在芯片外部加电容；三是加入选通信号。

3.【简答题】【嵌入式硬件基础与实践】解释 SRAM、SSRAM、SDRAM 三个名词？

SRAM：静态随机存取存储器（Static Random-Access Memory，SRAM）是随机存取存储器的一种。

SSRAM：Synchronous Static Random Access Memory 的缩写，即同步静态随机存取存储器。

SDRAM：同步动态随机存取内存（synchronous dynamic random-access memory，简称 SDRAM）是有一个同步接口的动态随机存取内存（DRAM）。

SSRAM 的所有访问都在时钟的上升/下降沿启动。地址、数据输入和其它控制信号均与时钟信号相关。

这一点与异步 SRAM 不同，异步 SRAM 的访问独立于时钟，数据输入和输出都由地址的变化控制。

SDRAM：Synchronous DRAM 同步动态随机存储器。

4.【简答题】【嵌入式硬件基础与实践】FPGA 和 ASIC 的概念

FPGA 是可编程 ASIC。

ASIC，专用集成电路，它是面向专门用途的电路，专门为一个用户设计和制造的。根据一个用户的特定要求，能以低研制成本，短、交货周期供货的全定制，半定制集成电路。

与门阵列等其它 ASIC（Application Specific IC）相比，它们又具有设计开发周期短、设计制造成本低、开发工具先进、标准产品无需测试、质量稳定以及可实时在线检验等优点。

5.【简答题】【嵌入式硬件基础与实践】单片机上电后没有运转，首先要检查什么？

首先应该确认电源电压是否正常。用电压表测量接地引脚跟电源引脚之间的电压，看是否是电源电压，例如常用的 5V。接下来就是检查复位引脚电压 是否正常。分别测量按下复位按钮和放开复位按钮的电压值，看是否正确。然后再检查晶振是否起振了，一般用示波器来看晶振引脚的波形，注意应该使用示波器探头的“ X10”档。另一个办法是测量复位状态下的 IO 口电平，按住复位键不放，然后测量 IO 口(没接外部上拉的 P0 口除外) 的电压，看是否是高电平，如果不是高电平，则多半是因为晶振没有起振。另外还要注意的地方是，如果使用片内 ROM 的话(大部分情况下如此，现在 已经很少有用外部扩 ROM 的了)，一定要将 EA 引脚拉高，否则会出现程序乱跑的情况。如果系统不稳定的话，有时是因为电源滤波不好导致的。在单片机的电源引脚跟地引脚之间接上一个 0.1uF 的电容会有所改善。如果电源没有滤波电容的话， 则需要再接一个更大滤波电容，例如 220uF 的。遇到系统不稳定时，就可以并上电容试试 (越靠近芯片越好)。

6.【简答题】【嵌入式硬件基础与实践】什么是同步逻辑和异步逻辑？

同步逻辑是时钟之间有固定的因果关系。异步逻辑是各时钟之间没有固定的因果关系。

常用逻辑电平：12V，5V，3.3V。

TTL 和 CMOS 不可以直接互连，由于 TTL 是在 0.3-3.6V 之间，而 CMOS 则是有在 12V 的有在 5V 的。CMOS 输出接到 TTL 是可以直接互连。TTL 接到 CMOS 需要在输出端口加一上拉电阻接到 5V 或者 12V。

7.【简答题】【嵌入式硬件基础与实践】如何解决亚稳态？

亚稳态是指触发器无法在某个规定时间段内达到一个可确认的状态。当一个触发器进入亚稳态时，既无法预测该单元的输出电平，也无法预测何时输出才能稳定在某个正确的电平上。

在亚稳态期间，触发器输出一些中间级电平，或者可能处于振荡状态，并且这种无用的输出电平可以沿信号通道上的各个触发器级联式传播下去。

解决方法主要有：

降低系统时钟；

用反应更快的触发器（FF），锁存器（LATCH）；

引入同步机制，防止亚稳态传播；

改善时钟质量，用边沿变化快速的时钟信号；

使用工艺好、时钟周期裕量大的器件。

8.【简答题】【嵌入式硬件基础与实践】锁存器、触发器、寄存器三者的区别？

触发器：能够存储一位二值信号的基本单元电路统称为“触发器”。

锁存器：一位触发器只能传送或存储一位数据，而在实际工作中往往希望一次传送或存储多位数据。为此可把多个触发器的时钟输入端 CP 连接起来，用一个公共的控制信号来控制，而各个数据端口仍然是各处独立地接收数据。这样所构成的能一次传送或存储多位数据的电路就称为“锁存器”。

寄存器：在实际的数字系统中，通常把能够用来存储一组二进制代码的同步时序逻辑电路称为寄存器。由于触发器内有记忆功能，因此利用触发器可以方便地构成寄存器。由于一个触发器能够存储一位二进制码，所以把 n 个触发器的时钟端口连接起来就能构成一个存储 n 位二进制码的寄存器。

区别：从寄存数据的角度来看，寄存器和锁存器的功能是相同的，它们的区别在于寄存器是同步时钟控制，而锁存器是电位信号控制。

可见，寄存器和锁存器具有不同的应用场合，取决于控制方式以及控制信号和数据信号之间的时间关系：若数据信号有效一定滞后于控制信号有效，则只能使用锁存器；若数据信号提前于控制信号到达并且要求同步操作，则可用寄存器来存放数据。

9.【简答题】【嵌入式硬件基础与实践】IC 设计中同步复位与异步复位的区别？

异步复位是不受时钟影响的，在一个芯片系统初始化（或者说上电）的时候需要这么一个全局的信号来对整个芯片进行整体的复位，到一个初始的确定状态。而同步复位需要在时钟沿来临的时候才会对整个系统进行复位。

10.【简答题】【嵌入式硬件基础与实践】多时域设计中，如何处理信号跨时域？

不同的时钟域之间信号通信时需要进行同步处理，这样可以防止新时钟域中第一级触发器的亚稳态信号对下级逻辑造成影响，其中对于单个控制信号可以用两级同步器，如电平、边沿检测和脉冲，对多位信号可以用 FIFO、

双口 RAM、握手信号等。

跨时域的信号要经过同步器同步，防止亚稳态传播。例如：时钟域 1 中的一个信号，要送到时钟域 2，那么在这个信号送到时钟域 2 之前，要先经过时钟域 2 的同步器同步后，才能进入时钟域 2。

这个同步器就是两级 d 触发器，其时钟为时钟域 2 的时钟。这样做是怕时钟域 1 中的这个信号，可能不满足时钟域 2 中触发器的建立保持时间，而产生亚稳态加粗样式，因为它们之间没有必然关系，是异步的。

这样做只能防止亚稳态传播，但不能保证采进来的数据的正确性。所以通常只同步很少位数的信号。比如控制信号，或地址。当同步的是地址时，一般该地址应采用格雷码，因为格雷码每次只变一位，相当于每次只有一个同步器在起作用，这样可以降低出错概率，象异步 FIFO 的设计中，比较读写地址的大小时，就是用这种方法。

如果两个时钟域之间传送大量的数据，可以用异步 FIFO 来解决问题。

我们可以在跨越 ClockDomain 时加上一个低电平使能的 LockupLatch 以确保 Timing 能正确无误。

11.【简答题】【嵌入式硬件基础与实践】说说静态、动态时序模拟的优缺点？

静态时序分析是采用穷尽分析方法来提取出整个电路存在的所有时序路径，计算信号在这些路径上的传播延时，检查信号的建立和保持时间是否满足时序要求，通过对最大路径延时和最小路径延时的分析，找出违背时序约束的错误。

它不需要输入向量就能穷尽所有的路径，且运行速度很快、占用内存较少，不仅可以对芯片设计进行全面的时序功能检查，而且还可利用时序分析的结果来优化设计，因此静态时序分析已经越来越多地被用到数字集成电路设计的验证中。

动态时序模拟就是通常的仿真，因为不可能产生完备的测试向量，覆盖门级网表中的每一条路径。因此在动态时序分析中，无法暴露一些路径上可能存在的时序问题。

12.【简答题】【嵌入式硬件基础与实践】什么是锁相环（PLL）？锁相环的工作原理是什么？

锁相环是一种反馈电路，其作用是使得电路上的时钟和某一外部时钟的相位同步。

PLL 通过比较外部信号的相位和由压控晶振（VCXO）的相位来实现同步的，在比较的过程中，锁相环电路会不断根据外部信号的相位来调整本地晶振的时钟相位，直到两个信号的相位同步。

在数据采集系统中，锁相环是一种非常有用的同步技术，因为通过锁相环，可以使得不同的数据采集板卡共享同一个采样时钟。

因此，所有板卡上各自的本地 80MHz 和 20MHz 时基的相位都是同步的，从而采样时钟也是同步的。

因为每块板卡的采样时钟都是同步的，所以都能严格地在同一时刻进行数据采集。

13.【简答题】【嵌入式硬件基础与实践】基本放大电路的种类及优缺点，广泛采用差分结构的原因？

基本放大电路按其接法的不同可以分为共发射极放大电路、共基极放大电路和共集电极放大电路，简称共基、共射、共集放大电路。

共射放大电路既能放大电流又能放大电压，输入电阻在三种电路中居中，输出电阻较大，频带较窄。常做为低频电压放大电路的单元电路。

共基放大电路只能放大电压不能放大电流，输入电阻小，电压放大倍数和输出电阻与共射放大电路相当，频率特性是三种接法中最好的电路。常用于宽频带放大电路。

共集放大电路只能放大电流不能放大电压，是三种接法中输入电阻最大、输出电阻最小的电路，并具有电压跟随的特点。常用于电压放大电路的输入级和输出级，在功率放大电路中也常采用射极输出的形式。

共集放大电路只能放大电流不能放大电压，是三种接法中输入电阻最大、输出电阻最小的电路，并具有电压跟随的特点。常用于电压放大电路的输入级和输出级，在功率放大电路中也常采用射极输出的形式。

14.【简答题】【嵌入式硬件基础与实践】异步清零和同步清零的区别

“异步”输入信号和时钟信号无关，是指输入信号变为有效状态，器件的状态就改变；“同步”输入信号和时钟信号有关，实际上输入信号和时钟信号进行了与运算或者与非运算，输入信号和时钟信号的运算结果是有效的，器件的状态才会改变。

15.【简答题】【嵌入式硬件基础与实践】有源滤波器和无源滤波器的区别

无源滤波器：这种电路主要有无源元件 R、L 和 C 组成；有源滤波器：集成运放和 R、C 组成，具有不用电感、体积小、重量轻等优点。 集成运放的开环电压增益和输入阻抗均很高，输出电阻小，构成有源滤波电路后还具有一定的电压放大和缓冲作用。但 集成运放带宽有限 ，所以目前的有源滤波电路的工作频率难以做得很高。

16.【简答题】【嵌入式硬件基础与实践】IIC 原理？

物理结构上，IIC 系统由一条串行数据线 SDA 和一条串行时钟线 SCL 组成。主机按一定的通信协议向从机寻址和进行信息 传输。在数据传输时，由主机初始化一次数据传输，主机使数据在 SDA 线上传输的同时还通过 SCL 线传输时钟。信息传输的对象和方向以及信息传输的开始和终 止均由主机决定。

每个器件都有一个唯一的地址，而且可以是单接收的器件（例如：LCD 驱动器）或者可以接收也可以发送的器件（例如：存储器）。发送器或接收器可以在主模式或从模式下操作，这取决于芯片是否必须启动数据的传输还是仅仅被寻址。

17.【简答题】【嵌入式硬件基础与实践】Bootloader 的启动方式？

（1）网络启动方式

这种方式的开发板不需要较大的存储介质，跟无盘工作站有点类似，但是使用这种启动方式之前，需要 Bootloader 安装到板上的 EPROM 或者 Flash 中。Bootloader 通过以太网接口远程下载 Linux 内核映像或者文件系统。Bootloader 下载文件一般都使用 TFTP 网络协议，还可以通过 DHCP 的方式动态配置 IP 地址。

（2）硬盘启动方式

传统的 Linux 系统运行在台式机或者服务器上，这些计算机一般都使用 BIOS 引导，并使用磁盘作为存储介质。Linux 传统上是 LILO (Linux Loader) 引导，后来又出现了 GUN 的软件 (Grand Unified Bootloader) 。这两种 Bootloader 广泛应用在 X86 的 Linux 系统上。

（3）Flash 启动方式

大多数嵌入式系统上都使用 Flash 存储介质。Flash 有很多类型，包括 NOR Flash、NAND Flash 和其它半导体盘。它们之间的不同在于： NOR Flash 支持芯片内执行（XIP， eXecute In Place），这样代码可以在 Flash 上直接执行而不必拷贝到 RAM 中去执行。而 NAND Flash 并不支持 XIP，所以要想执行 NAND Flash 上的代码，必须先将其拷贝到 RAM 中去，然后跳到 RAM 中去执行。NOR Flash 使用最为普遍。Bootloader 一般放在 Flash 的底端或者顶端，这需要根据处理器的复位向量来进行设置。可以配置成 MTD 设备来访问 Flash 分区。

18.【简答题】【嵌入式硬件基础与实践】DMA 原理？

一个完整的 DMA 传输过程必须经过下面的 4 个步骤。

（1）DMA 请求：CPU 对 DMA 控制器初始化，并向 I/O 接口发出操作命令，I/O 接口提出 DMA 请求。

（2）DMA 响应：DMA 控制器对 DMA 请求判别优先级及屏蔽，向总线裁决逻辑提出总线请求。当 CPU 执行完当前总线周期即可释放总线控制权。此时，总线裁决逻辑输出总线应答，表示 DMA 已经响应，通过 DMA 控制器通知 I/O 接口开始 DMA 传输。

（3）DMA 传输：DMA 控制器获得总线控制权后，CPU 即刻挂起或只执行内部操作，由 DMA 控制器输出读写命令，直接控制 RAM 与 I/O 接口进行 DMA 传输。

（4）DMA 结束：当完成规定的成批数据传送后，DMA 控制器即释放总线控制权，并向 I/O 接口发出结束信号。当 I/O 接口收到结束信号后，一方面停 止 I/O 设备的工作，另一方面向 CPU 提出中断请求，使 CPU 从不介入的状态解脱，并执行一段检查本次 DMA 传输操作正确性的代码。最后，带着本次操作结果及状态继续执行原来的程序。

由此可见，DMA 传输方式无需 CPU 直接控制传输，也没有中断处理方式那样保留现场和恢复现场的过程，通过硬件为 RAM 与 I/O 设备开辟一条直接传送数据的通路，使 CPU 的效率大为提高。

19.【简答题】【嵌入式硬件基础与实践】中断的上半段和下半段指什么？

中断是一个很霸道的东西，处理器一旦接收到中断，就会打断正在执行的代码，调用中断处理函数。如果在中

断处理函数中没有禁止中断，该中断处理函数执行过程中仍有可能被其他中断打断。出于这样的原因，大家都希望中断处理函数执行得越快越好。

另外，中断上下文中不能阻塞，这也限制了中断上下文中能干的事。

基于上面的原因，内核将整个的中断处理流程分为了上半部和下半部。上半部就是之前所说的中断处理函数，它能最快的响应中断，并且做一些必须在中断响应之后马上要做的事情。而一些需要在中断处理函数后继续执行的操作，内核建议把它放在下半部执行。

拿网卡来举例，在 linux 内核中，当网卡一旦接受到数据，网卡会通过中断告诉内核处理数据，内核会在网卡中断处理函数（上半部）执行一些网卡硬件的必要设置，因为这是在中断响应后急切要干的事情。接着，内核调用对应的下半部函数来处理网卡接收到的数据，因为数据处理没必要在中断处理函数里面马上执行，可以将中断让出来做更紧迫的事情。

20.【简答题】【嵌入式硬件基础与实践】can 原理？

控制器局域网总线(CAN，Controller Area Network)是一种用于实时应用的串行通讯协议总线，它可以使用双绞线来传输信号，是世界上应用最广泛的现场总线之一。CAN 协议由德国的 Robert Bosch 公司开发，用于汽车中各种不同元件之间的通信，以此取代昂贵而笨重的配电线束。该协议的健壮性使其用途延伸到其他自动化和工业应用。CAN 协议的特性包括完整性的串行数据通讯、提供实时支持、传输速率高达 1Mb/s、同时具有 11 位的寻址以及检错能力。

CAN 总线是一种多主方式的串行通讯总线，基本设计规范要求有高的位速率，高抗电子干扰性，并且能够检测出产生的任何错误。CAN 总线可以应用于汽车电控制系统、电梯控制系统、安全监测系统、医疗仪器、纺织机械、船舶运输等领域。

CAN 总线的特点

（1）具有实时性强、传输距离较远、抗电磁干扰能力强、成本低等优点;

（2）采用双线串行通信方式，检错能力强，可在高噪声干扰环境中工作;

（3）具有优先权和仲裁功能，多个控制模块通过 CAN 控制器挂到 CAN-bus 上，形成多主机局部网络;

（4）可根据报文的 ID 决定接收或屏蔽该报文;

（5）可靠的错误处理和检错机制;

（6）发送的信息遭到破坏后，可自动重发;

（7）节点在错误严重的情况下具有自动退出总线的功能;

（8）报文不包含源地址或目标地址，仅用标志符来指示功能信息、优先级信息。

21.【简答题】【嵌入式硬件基础与实践】负反馈种类及优点？

电压并联反馈，电流串联反馈，电压串联反馈和电流并联反馈。降低放大器的增益灵敏度，改变输入电阻和输出电阻，改善放大器的线性和非线性失真，有效地扩展，放大器的通频带，自动调节作用。

22.【简答题】【嵌入式硬件基础与实践】STM32F1 和 F4 的区别？

内核不同：F1 是 ARM M3 内核，F4 是 ARM M4 内核；

主频不同：F1 主频 72MHz，F4 主频 168MHz；

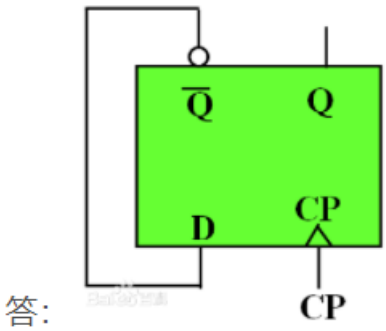
浮点运算：F1 无浮点运算单位，F4 有；

功能性能：F4 外设比 F1 丰富且功能更强大，比如 GPIO 翻转速率、上下拉电阻配置、ADC 精度等；

内存大小：F1 内部 SRAM 最大 64K，F4 有 192K(112+64+16)。

23.【简答题】【嵌入式硬件基础与实践】请画出用 D 触发器实现 2 倍分频的逻辑电路

答：



24.【简答题】【嵌入式硬件基础与实践】你知道那些常用逻辑电平？ TTL 与 COMS 电平可以直接互连吗？

常用的电平标准，低速的有 RS232、RS485 、RS422、 TTL、CMOS 、LVTTL、 LVCMOS、ECL 、ECL、LVPECL 等，高速的有 LVDS、 GTL、PGTL 、 CML、 HSTL、SSTL 等。一般说来，CMOS 电平比 TTL 电平有着更高的噪声容限。 如果不考虑速度和性能，一般 TTL 与 CMOS 器件可以互换。但是需要注意有时候负载效应可能引起电路工作不正常，因为有些 TTL 电路需要下一级的输入阻抗作为负载才能 正常工作。

25.【简答题】【嵌入式硬件基础与实践】什么是频率响应，怎么才算是稳定的频率响应，简述改变频率响应曲线的几个方法？

这里仅对放大电路的频率响应进行说明。 在放大电路中，由于电抗元件 (如电容、电感线圈等)及晶体管极间电容的存在，当输入信号的频率过低或过高时，放大电路的放大倍数的数值均会降低，而且还将产生相位超前或之后现象。也就是说，放大电路的放大倍数 (或者称为增 益)和输入信号频率是一种函数关系，我们就把这种函数关系成为放大电路的频 率响应或频率特性。

放大电路的频率响应可以用幅频特性曲线和相频特性曲线来描述，如果一个 放大电路的幅频特性曲线是一条平

行于 x 轴的直线(或在关心的频率范围内平行于 x 轴)，而相频特性曲线是一条通过原点的直线 (或在关心的频率范围是条通过原点的直线)，那么该频率响应就是稳定的。

改变频率响应的方法主要有： (1) 改变放大电路的元器件参数； (2) 引入新的元器件来改善现有放大电路的频率响应； (3) 在原有放大电路上串联新的放大电路构成多级放大电路。

26.【简答题】【交互设计】不同平台交互设计的区别？

（1）操作方式不同

Web： 鼠标滑过可收起展开；

App：左右滑动；重要元素放在单手点击热区内；图标提示可以点击

（2）设备尺寸不同

Web ：浏览器分辨率和布局；窗口尺寸可变，响应式设计

App：分辨率不同，适配；尺寸小，信息分优先级展示；横屏

（3）使用环境

Web：使用时间长，网络稳定，用户更专注

App：碎片化时间，提供“稍后阅读”,“收藏”等功能；误操作可能性加大，防错，提供恢复错误；网络异常提示；耗流量较多的操作提示用户，经过允许再操作。

（4）通知方式

Web：用户不怎么使用浏览器的通知中心

App：推送

（5）位置服务

Web：定位获取当前城市

App：精确的位置。利用位置提供服务，如生活服务类可以查询我的位置附近的美食、商场、影院，旅行类提供机场附近酒店、饭馆信息。

27.【简答题】【交互设计】你如何理解“交互设计”和“用户体验”?它们是什么关系?

首先，交互设计可以理解为是用户与产品或某个功能交互的一个过程，比如包括交互前的引导，交互中的提示，交互后的反馈。更重要的是框架层级的划分。交互设计更偏重于执行层面。而用户体验其实拥有更广的范围，用户体验里包含了交互体验、视觉体验、流畅度体验、用户感知等。可以说用户体验只是一个理念。交互设计和用户体验的关系其实就是用户体验包涵的交互的体验，而反过来说交互设计的过程中要运用用户体验的理念。

28.【简答题】【交互设计】MD 和 iOS 规范的区别？

MD 注重视觉效果的规范统一；iOS 注重功能表意清晰，操作明确，注重应用与系统间功能的一致性。

29.【简答题】【交互设计】static 全局变量与普通的全局变量有什么区别？static 函数与普通函数有什么区别？

全局变量(外部变量)的说明之前再冠以 static 就构成了静态的全局变量。全局变量本身就是静态存储方式，静态全局变量当然也是静态存储方式。这两者在存储方式上并无不同。这两者的区别虽在于非静态全局变量的作用域是整个源程序， 当一个源程序由多个源文件组成时，非静态的全局变量在各个源文件中都是有效的。而静态全局变量则限制了其作用域，即只在定义该变量的源文件内有效， 在同一源程序的其它源文件中不能使用它。由于静态全局变量的作用域局限于一个源文件内，只能为该源文件内的函数公用，因此可以避免在其它源文件中引起错误。从以上分析可以看出，把局部变量改变为静态变量后是改变了它的存储方式即改变了它的生存期。把全局变量改变为静态变量后是改变了它的作用域，限制了它的使用范围。static 函数与普通函数作用域不同。仅在本文件。只在当前源文件中使用的函数应该说明为内部函数(static)，内部函数应该在当前源文件中说明和定义。对于可在当前源文件以外使用的函数，应该在一个头文件中说明，要使用这些函数的源文件要包含这个头文件。

30.【简答题】【交互设计】请问以下代码有什么问题：

```
int main() {
char a;
char *str=&a;
strcpy(str,"hello");
printf(str);
return 0;
}
```

没有为 str 分配内存空间，将会发生异常，问题出在将一个字符串复制进一个字符变量指针所指地址。虽然可以正确输出结果，但因为越界进行内在读写而导致程序崩溃。

31.【简答题】【交互设计】什么是预编译，何时需要预编译？

①预编译又称为预处理,是做些代码文本的替换工作。处理#开头的指令,比如拷贝#include 包含的文件代码，#define 宏定义的替换,条件编译等，就是为编译做的预备工作的阶段，主要处理#开始的预编译指令，预编译指令指示了在程序正式编译前就由编译器进行的操作，可以放在程序中的任何位置。

②总是使用不经常改动的大型代码体。程序由多个模块组成，所有模块都使用一组标准的包含文件和相同的编译选项。在这种情况下，可以将所有包含文件预编译为一个预编译头。

32.【简答题】【交互设计】结构与联合有和区别？

(1). 结构和联合都是由多个不同的数据类型成员组成, 但在任何同一时刻, 联合中只存放了一个被选中的成员（所有成员共用一块地址空间）, 而结构的所有成员都存在（不同成员的存放地址不同）。

(2). 对于联合的不同成员赋值, 将会对其它成员重写, 原来成员的值就不存在了, 而对于结构的不同成员赋值是互不影响的

33.【简答题】【交互设计】描述内存分配方式以及它们的区别?

1） 从静态存储区域分配。内存在程序编译的时候就已经分配好，这块内存在程序的整个运行期间都存在。例如全局变量，static 变量。

2） 在栈上创建。在执行函数时，函数内局部变量的存储单元都可以在栈上创建，函数执行结束时这些存储单元自动被释放。栈内存分配运算内置于处理器的指令集。

3） 从堆上分配，亦称动态内存分配。程序在运行的时候用 malloc 或 new 申请任意多少的内存，程序员自己负责在何时用 free 或 delete 释放内存。动态内存的生存期由程序员决定，使用非常灵活，但问题也最多

34.【简答题】【交互设计】请说出 const 与#define 相比，有何优点？

Const 作用：定义常量、修饰函数参数、修饰函数返回值三个作用。被 Const 修饰的东西都受到强制保护，可以预防意外的变动，能提高程序的健壮性。

1） const 常量有数据类型，而宏常量没有数据类型。编译器可以对前者进行类型安全检查。而对后者只进行字符替换，没有类型安全检查，并且在字符替换可能会产生意料不到的错误。

2） 有些集成化的调试工具可以对 const 常量进行调试，但是不能对宏常量进行调试。

35.【简答题】【交互设计】简述数组与指针的区别？

数组要么在静态存储区被创建（如全局数组），要么在栈上被创建。指针可以随时指向任意类型的内存块。

(1)修改内容上的差别

char a[] = “hello”;

a[0] = ‘X’;

char *p = “world”; // 注意 p 指向常量字符串

p[0] = ‘X’; // 编译器不能发现该错误，运行时错误

(2) 用运算符 sizeof 可以计算出数组的容量（字节数）。sizeof§,p 为指针得到的是一个 指针变量的字节数，而不是 p 所指的内存容量。C++/C 语言没有办法知道指针所指的内存容量，除非在申请内存时记住它。注意当数组作为函数的参数进行传递时，该数组自动退化为同类型的指针。

char a[] = “hello world”;

char *p = a;

cout<< sizeof(a) << endl; // 12 字节

cout<< sizeof § << endl; // 4 字节

计算数组和指针的内存容量

void Func(char a[100])

{undefined

cout<< sizeof(a) << endl; // 4 字节而不是 100 字节}

36.【简答题】【交互设计】一个单向链表，不知道头节点,一个指针指向其中的一个节点，问如何删除这个指针指向的节点？

将这个指针指向的 next 节点值 copy 到本节点，将 next 指向 next->next,并随后删除原 next 指向的节点。

37.【简答题】【交互设计】中断与异常的区别

中断是外部硬件产生的电信号通过处理器的中断端口打断处理器的处理过程异常是处理器内部执行到错误指令、或者在执行期间出现错误，必须靠内核处理的时候就会产生一个异常。

38.【简答题】【交互设计】堆栈溢出一般是由什么原因导致的?

1.没有回收垃圾资源

2.层次太深的递归调用

39.【简答题】【交互设计】局部变量能否和全局变量重名?

能，局部会屏蔽全局。要用全局变量，需要使用”::”局部变量可以与全局变量同名，在函数内引用这个变量时，会用到同名的局部变量，而不会用到全局变量。对于有些编译器而言，在同一个函数内可以定义多个同名的局部变量，比如在两个循环体内都定义一个同名的局部变量，而那个局部变量的作用域就在那个循环体内

40.【简答题】【交互设计】如何引用一个已经定义过的全局变量?

可以用引用头文件的方式，也可以用 extern 关键字，如果用引用头文件方式来引用某个在头文件中声明的全局变量，假定你将那个变量写错了，那么在编译期间会报错，如果你用 extern 方式引用时，假定你犯了同样的错误，那么在编译期间不会报错，而在连接期间报错。

41.【简答题】【交互设计】下面的代码输出是什么，为什么？

```
void foo(void)
{
unsigned int a = 6;int b = -20;
(a+b > 6) ? puts("> 6"): puts("<= 6");
}
```

这个问题测试你是否懂得 C 语言中的整数自动转换原则，我发现有些开发者懂得极少这些东西。不管如何，这无符号整型问题的答案是输出是">6”。原因是当表达式中存在有符号类型和无符号类型时所有的操作数都自动转换为无符号类型。因此-20 变成了一个非常大的正整数，所以该表达式计算出的结果大于 6。

42.【编程题】【交互设计】用户输入 M,N 值，从 1 至 N 开始顺序循环数数，每数到 M 输出该数值，直至全部输出。写出 C 程序。循环链表，用取余操作做。

```
#include <stdio.h>
#include "stdliB、h"
#define NULL 0
#define TYPE struct stu
#define LEN sizeof (struct stu)

struct stu
{
int data;
struct stu*next;
};

TYPE *line(int n)
{
int sum=1;
struct stu *head,*pf,*pb;
int i;
for(i=0;i<n;i++)
{
pb=(TYPE*) malloc(LEN);
pb->data=i+1;
```

```
if (i==0)
pf=head=pb;
else
pf->next=pb;
if (i==(n-1))
pb->next=head;
else pb->next=NULL;
pf =pb;
sum++;
}
return(head);
}
display(TYPE*L)
{
int i;
struct stu *head,*pf;
pf=L;
head=L;
printf("%din",pf->data);pf=pf->next;
for(i=0;pf!-head;i++)
{
printf("%dn",pf->data);
pf =pf->next;
}
}

main()
{
int M,N,x,i;
struct stu*p,*q;

printf( "please scanf M and N (M<N)");
scanf("%d %d",&M,&N);
p=line(N);
display(p);
        x=N;
```

```
while(x)
{
for(i=1;i<M-1;i++)
{
p p->next;
}
printf("p.data=%din",p->data);//找到了 M 前面的 1 个
q=p->next;
printf("%dn",q->data) ;//输出 M
p->next = p->next->next;//释放 M
p=p->next;//从 M 的下一个又开始数
free(q) ;
X--;
}
getch();
}
```

43.【编程题】【交互设计】给定一个整型变量 a，写两段代码，第一个设置 a 的 bit3，第二个清除 a 的 bit，在以上两个操作中，要保持其它位不变。

```
define BIT3 (0x1<<3)
static int a;
void set_bit3(void)
{
    a |= BIT3;
}

void clear_bit3(void)
{
     a &= ~BIT3;
}
```

44.【编程题】【交互设计】输入某年某月某日，判断这一天是这一年的第几天?

程序分析：以 3 月 5 日为例，应该先把前两个月的加起来，然后再加上 5 天即本年的第几天，特殊情况，闰年且输入月份大于 3 时需考虑多加一天

```
main()
{ int day,month,year,sum,leap;
printf(“\nplease input year,month,day\n”);
scanf(“%d,%d,%d”,&year,&month,&day);
switch(month)/*先计算某月以前月份的总天数*/
{
case 1:sum=0;break;
case 2:sum=31;break;
case 3:sum=59;break;
case 4:sum=90;break;
case 5:sum=120;break;
case 6:sum=151;break;
case 7:sum=181;break;
case 8:sum=212;break;
case 9:sum=243;break;
case 10:sum=273;break;
case 11:sum=304;break;
case 12:sum=334;break;
default:printf(“data error”);break;
}
sum=sum+day; /*再加上某天的天数*/
if(year%400==0||(year%4==0&&year%100!=0))/*判断是不是闰年*/
leap=1;
else
leap=0;
if(leap==1&&month>2)/*如果是闰年且月份大于 2,总天数应该加一天*/
sum++;
printf(“It is the %dth day.”,sum);}
```

45.【编程题】【交互设计】要求输出国际象棋棋盘。1.程序分析：用 i 控制行，j 来控制列，根据 i+j 的和的变化来控制输出黑方格，还是白方格。

```
#include “stdio.h”
main()
{ int i,j;
for(i=0;i<8;i++)
```

```
{
for(j=0;j<8;j++)
if((i+j)%2==0)
printf(“%c%c”,219,219);
else
printf(” “);
printf(“\n”);
}
}
```

46.【编程题】【交互设计】打印楼梯，同时在楼梯上方打印两个笑脸。

程序分析：用 i 控制行，j 来控制列，j 根据 i 的变化来控制输出黑方格的个数。

```
#include “stdio.h”
main()
{ int i,j;
printf(“\1\1\n”);/*输出两个笑脸*/
for(i=1;i<11;i++)
{
for(j=1;j<=i;j++)
printf(“%c%c”,219,219);
printf(“\n”);
}
}
```

47.【编程题】【交互设计】有一对兔子，从出生后第 3 个月起每个月都生一对兔子，小兔子长到第三个月后每个月又生一对兔子，假如兔子都不死，问每个月的兔子总数为多少?

程序分析： 兔子的规律为数列 1,1,2,3,5,8,13,21….

```
main()
{ long f1,f2;
int i;
f1=f2=1;
```

```
for(i=1;i<=20;i++)
{ printf(“%12ld %12ld”,f1,f2);
if(i%2==0) printf(“\n”);/*控制输出，每行四个*/
f1=f1+f2; /*前两个月加起来赋值给第三个月*/
f2=f1+f2; /*前两个月加起来赋值给第三个月*/
}
}
```

48.【编程题】【交互设计】判断 101-200 之间有多少个素数，并输出所有素数。

```
#include “math.h”
main()
{
int m,i,k,h=0,leap=1;
printf(“\n”);
for(m=101;m<=200;m++)
{ k=sqrt(m+1);
for(i=2;i<=k;i++)
if(m%i==0)
{leap=0;break;}
if(leap) {printf(“%-4d”,m);h++;
if(h%10==0)
printf(“\n”);
}
leap=1;
}
printf(“\nThe total is %d”,h);
}
```

49.【编程题】【交互设计】求 s=a+aa+aaa+aaaa+aa…a 的值，其中 a 是一个数字。例如 2+22+222+2222+22222(此时共有 5 个数相加)，几个数相加有键盘控制。

```
main()
{
int a,n,count=1;
long int sn=0,tn=0;
```

```
printf(“please input a and n\n”);
scanf(“%d,%d”,&a,&n);
printf(“a=%d,n=%d\n”,a,n);
while(count<=n)
{
tn=tn+a;
sn=sn+tn;
a=a*10;
++count;
}
printf(“a+aa+…=%ld\n”,sn);
}
```

50.【编程题】【交互设计】一个数如果恰好等于它的因子之和，这个数就称为“完数”。例如 6=1＋2＋3.编程找出 1000 以内的所有完数。

```
main()
{
static int k[10];
int i,j,n,s;
for(j=2;j<1000;j++)
{
n=-1;
s=j;
for(i=1;i<j;i++)
{
if((j%i)==0)
{ n++;
s=s-i;
k[n]=i;
}
}
if(s==0)
{
printf(“%d is a wanshu”,j);
for(i=0;i<N;I++)
```

```
printf(“%d,”,k[i]);
printf(“%d\n”,k[n]);
}
}
}
```

51.【简答题】【技术评估与设计】a=5;b=6; a+=b++;printf（"a=‰d，b=%d"）;最终结果是多少（）

a=11,b=6

a=12,b=6

a=11,b=7

D、a=12,b=7

答案：C

解析：因为 a+=b++ 是先算 a+b 然后 b 再自增。

52.【简答题】【技术评估与设计】嵌入式产品产生的成本大致分为那 2 类？

线上系统开发成本；

线下运营支撑成本。

53.【简答题】【技术评估与设计】请简述全数字模拟测试。

全数字模拟测试是指采用数学平台的方法，将嵌入式软件从系统中剥离出来，通过开发 CPU 指令、常用芯片、I/O、中断、时钟等模拟器在开发主机平台（Host）上实现嵌入式软件的测试。该方法操作简单，适用于功能测试，是一种可以借鉴的常规软件测试方法。但是全数字模拟测试有较大的局限性，使用不同语言编写的嵌入式软件需要不同的仿真程序来执行，通用性差，实时性与准确性难以反映出嵌入式软件的真实情况，当并发事件要求一定的同步关系时，维护统一、精确地系统时钟，理顺时序关系相当困难。因此，设计一个能进行系统测试的环境代价太大，全数字模拟测试只能作为嵌入式软件测试的辅助手段。

54.【简答题】【技术评估与设计】嵌入式软件测试的测试内容包括那几个部分。

1.单元测试；2.集成测试；3.确认测试；4.系统测试。

55.【简答题】【技术评估与设计】嵌入式产品开发结束后为什么要进行测试。

测试能提高和改善嵌入式软件的质量

56.【简答题】【技术评估与设计】硬件开发流程对硬件开发的全过程进行了科学分解，规范了硬件开发的那五大任务？

硬件需求分析及总体方案制定

单板设计方案及单板详细设计

原理图设计及 PCB 设计

调试及验收

开发文档规范及归档要求

57.【简答题】【技术评估与设计】硬件总体设计方案主要有下列内容：

系统功能及功能指标

系统总体结构图及功能划分

单板命名

系统逻辑框图

组成系统各功能块的逻辑框图，电路结构图及单板组成

单板逻辑框图和电路结构图

关键技术讨论

关键器件

58.【简答题】【技术评估与设计】单板设计方案主要包括下列内容

单板在整机中的的位置：单板功能描述

单板尺寸

单板逻辑图及各功能模块说明

单板软件功能描述

单板软件功能模块划分

接口定义及与相关板的关系

重要性能指标、功耗及采用标准

开发用仪器仪表等

59.【简答题】【技术评估与设计】请简述 PCB 的制作过程

1.绘制 PCB 的大小，如果是产品的改进，最好兼容以前的外部封装。（在 mechanical 或 keep out 层画板框）

2 确定 PCB 的层数，如在设置四层板时，内电层有正片和负片两种选择方式，像上、下两层的 signal 为正片，即在铺铜时，显示连接区域的铜片，负片正好相反。负片/正片电源分隔时，隔离距离应大于 1mm，即线宽需大于 1mm。

3PCB 规则的设置，设置好一个 PCB 规则，如线宽，间距，过孔大小，元器件的间距，高度等因素，当触犯该规则的限制条件时，电路会高亮、也可以导入上个项目使用的规则。

4 将元器件从原理图中导入 PCB 板，并进行模块化布局。

区域化布局选择器件：Tools -> Cross Select Mode（交互选择）

5 布线，先连接近点复杂的线路，在连接远点的线路：先近后远

6 铺铜、补泪滴进一步保证 PCB 的质量

7 丝印层注释，丝印层可对部分模块进行详细说明，有利于开发者的调试与使用，防止过孔部分覆盖到丝印。

60.【简答题】【技术评估与设计】嵌入式软件测试或叫交叉测试（Cross-test），测试内容包括：

1.单元测试；2.集成测试；3.确认测试；4.系统测试。

61.【简答题】【技术评估与设计】嵌入式软件做 FMEA 的步骤？

定义范围，结构分析，功能分析，风险分析，失效分析，优先。

62.【简答题】【技术评估与设计】嵌入式产品开发结束后为什么要进行测试。

测试能提高和改善嵌入式软件的质量

63.【简答题】【技术评估与设计】在这一个阶段，我们需要弄清楚的是产品的需求从何而来，一个成功的产品，我们需要满足哪些需求。只有需求明确了，我们的产品开发目标才能明确。在产品需求分析阶段，我们可以通过那些途径获取产品需求。

1）市场分析与调研，主要是看市场有什么需求,还有就是前沿的技术是什么（站在做一款产品的角度）；

2）客户调研和用户定位，从市场广大客户那获取最准确的产品需求（要注意分析市场，产品生命周期升级是否方便）；

3）利润导向(成本预算)；

4）如果是外包项目，则需要我们的客户提供产品的需求（直接从客户那获取，让客户签协议）；

64.【简答题】【技术评估与设计】产品调试与验证阶段是调整硬件或代码，修正其中存在的问题和 BUG，使之能正常运行，并尽量使产品的功能达到产品需求规格说明要求，请简单说明软件和硬件的调试和验证。

硬件部分：

1）目测加工会得 PCB 板是否存在短路，器件是否焊错，或漏焊接；

2）测试各电源对地电阻是否正常；

3）上电，测试电源是否正常；

4）分模块调试硬件模块，可借助示波器、逻辑分析仪等根据。

软件部分：

验证软件单个功能是否实现，验证软件整个产品功能是否实现。

65.【简答题】【技术评估与设计】一个嵌入式产品开发到了测试环节一般要经历那几种测试？

功能测试（测试不通过，可能是有 BUG）；

压力测试（测试不通过，可能是有 BUG 或哪里参数设计不合理）；

性能测试（产品性能参数要提炼出来，供将来客户参考，这个就是你的产品特征的一部分）；

其他专业测试：包括工业级的测试，例如含抗干扰测试，产品寿命测试，防潮湿测试，高温和低温测试（有的产品有很高的温度或很低的温度工作不正常，甚至停止工作）。

有的设备电子元器件在特殊温度下，参数就会异常，导致整个产品出现故障或失灵现象的出现；有的设备，零下几十度的情况下，根本就启动不了，开不了机；有的设备在高温下，电容或电阻值就会产生物理的变化，这些都会影响到产品的质量。这里要引出一个话题，工业级产品与消费类产品有什么区别呢？工业级的产品就要避免这些异常和特殊问题，有的产品是在很深的海里工作，或者在严寒的山洞工作，或者火热沙漠工作，或者颠簸的设备上，比如汽车；或者是需要防止雷击；所以这就是工业级产品跟消费类产品的区别，消费类的产品就不需要做这么多的测试。

66.【简答题】【技术评估与设计】开发一个嵌入式产品一般要对开发成本进行评估，那么基本评估方向有那些？

人力成本（开发人员、管理人员、销售人员、其他行政等辅助人员）的开销

材料（硬件物料和损耗，有时候需要投几次 PCB 版才把产品稳定下来）的开销

开发系统和开发工具软件的开销

硬件工具的开销（例如示波器、仿真器等）

67.【简答题】【技术评估与设计】黑盒白盒法测试属于哪一类测试，它有什么特点。

动态测试；动态测试时软件必须运行。为了较快得到测试效果，通常先进行功能测试，达到所有功能后，为确定软件的可靠性进行必要的覆盖测试。

在软件开发的不同时期进行动态测试，测试又分为单元测试、集成测试、确认测试、系统测试。

68.【简答题】【技术评估与设计】嵌入式软件做 FMEA 的步骤？

定义范围，结构分析，功能分析，风险分析，失效分析，优先。

69.【简答题】【技术评估与设计】单元测试方案之一要用到 IPL 公司的什么测试工具？

Cantata++

70.【简答题】【技术评估与设计】Cantata++测试工具的主要部分？

CTH 测试功能库，Cantata++通过 CTH 提供的测试函数执行测试，提供测试所需用例的输入输出，并检查输出结果是否符合要求，给出合格/不合格的确切结果。打桩、封装和动态分析的执行也是利用 CTH。

71.【简答题】【技术评估与设计】嵌入式软件系统测试具有那些特点。

1）测试软件功能依赖不需编码的硬件功能，快速定位软硬件错误困难；

2）强壮性测试、可知性测试很难编码实现；

3）交叉测试平台的测试用例、测试结果上载困难；

4）基于消息系统测试的复杂性，包括线程、任务、子系统之间的交互，并发、容错和对时间的要求；

5）性能测试、确定性能瓶颈困难；

6）实施测试自动化技术困难。大量统计资料表明，软件测试的工作量往往占软件开发总工作量的 40%以上，在极端情况，测试那种关系人的生命安全的重要的行业中的嵌入式软件所花费的成本，可能相当于软件工程其他开发步骤总成本的三倍到五倍。

72.【单选题】【语言概念】将递归算法转换为非递归算法通常需要使用（D）

A 栈

B 队列

C 队列

D 广义表

73.【单选题】【语言概念】选项中那一行代码可以替换//add code here 而不产生编译错误（）

```
public abstruct class MyClass{
    publicint testInt = 5;
    //addcode here
    publicvoid method(){
    }
}
```

A public abstruct void another Method(){}

B testInt = testInt * 5

C public int method();

D public abstruct void another Method(int a)

答案：D

解析：

A:该项方法有 abstract 修饰，所以是抽象方法，由于抽象方法不能有方法体，所以 A 项错误

B:类体中只能定义变量和方法，不能有其他语句，所以 B 项错误

C:选项中的方法和类中的方法重复，所以会发生编译异常，所以 C 项错误

74.【单选题】【语言概念】下列有关软链接表述正确的是？（）

A 不可以对不存在的文件创建软链接

B 不能对目录创建软链接

C 和普通件没有什么不同，inode 都指向同一个文件在硬量中的区块

D 保存了其代表的文件的绝对路径是另一种文件。在硬盘上有独立的区块，访问时替代自身路径

答案：C

解析：

A：错。后半句说的是硬链接。硬链接是共同拥有同一个 inode,不过是每个链接名不同，暂时理解成不同的文件名却指向同一文件。一个文件每加一个硬链接 linkcount 加 1。

B：错。可以对目录创建软链接

D：错。可以对不存在的文件创建软链接

75.【单选题】【语言概念】将两个各有 n 个元素的有序表归并成一个有序表，最少的比较次数是？（）

A n

B 2n

C n-1

D 2n-1

答案：A

解析：

归并排序是将两个或两个以上的有序子表合并成一个新的有序表。在归并排序中，核心步骤是将相邻的两个有序序列归并为一个有序序列。

题目中告诉我们，有两个各有 n 个元素的有序序列，要将这两个序列归并成一个有序序列，其方法是依次从小到大取每个序列中的元素进行比较，将较小的放进一个新的序列中，直到取完一个有序序列中的所有元素。再把另一个序列中剩下的元素放进新序列的后面即可。

最好的情况是一个有序序列中的最小元素大于另一个有序序列中的所有元素，这样只需要比较 n 次。

76.【单选题】【语言概念】有关 Java 静态初始化块说法不正确的是？（）

A 用户可以控制何时执行静态初始化块

B 无法直接词用静态初始化块

C 在创建第一个实例前，将自动调用静态初始化块来初始化

D 静态初始化块没有访问修饰符和参数

答案：A

解析：

JAVA 的初始化顺序：

父类的静态成员初始化>父类的静态代码块>子类的静态成员初始化>子类的静态代码块>父类的代码块>父类的构造方法>子类的代码块>子类的构造方法

77.【单选题】【语言概念】常见的 socket 类型中不包括下面哪项:

A、SOCK_STREAM

B、SOCK_DGRAM

C、SOCK_ DTRAN

D、SOCK_RAW

答案：C

解析：SOCKET_STREAM：TCP 协议

SOCKET_DGRAM:不连续不可靠的数据包连接

Socket_raw: 提供原始网络协议存取

78.【单选题】【语言概念】若有说明语句：char c=’\72’；则变量 c（）。

A、 包含 1 个字符 **B、 包含 2 个字符**

C、 包含 3 个字符 **D、 说明不合法，c 的值不确定**

答案：A

解析：char c; 声明它是一个 char ,只有 1 个字符 的 内存空间

所以不可能包含 2 个字符,也不可能 包含 3 个字符.b 和 c 可以排除了.'\72' -- ,字符常量 通常 用单引号括起来,所以单引号是对的,没有疑问。这里要记住,用反斜杠带数字,是八进制数,八进制数只能用到数字 0,1,2,3,4,5,6,7 如果出现 8,9 就不合法,现在 72 是 合法的。另外,字符常量 最大占 1 个字节,数值不能超出 ASCII 码最大值.八进制数 072 显然没超出.所以 排除了 D、

79.【单选题】【语言概念】若二维数组 a 有 m 列（假如 a[0][0]位于数组的第一个位置上）， 则计算任一元素 a[i][j] 在数组中位置的公式为（）

A、I*m+j **B、 j*p+i** **C、 i*m+j-1** **D、 i*m+j+1**

答案：A

解析: a[i][j]元素之前有 i 行元素(每行有 m 个元素)，在 a[i][j]的前面还有 j 个元素，因此 a[i][lj]之前共有 i * m+j 个元素。

80.【多选题】【语言概念】Servlet 的生命周期可以分为初始化阶段，运行阶段和销毁阶段三个阶段，以下过程属于初始化阶段是()。

A 加载 Servlet 类及.class 对应的数据

B 创建 serletRequest 和 servletResponse 对象

C 创建 ServletConfig 对象

D 创建 Servlet 对象

答案：ACD

解析：

Servlet 的生命周期一般可以用三个方法来表示：

init()：仅执行一次，负责在装载 Servlet 时初始化 Servlet 对象

service() ：核心方法，一般 HttpServlet 中会有 get,post 两种处理方式。在调用 doGet 和 doPost 方法时会构造 servletRequest 和 servletResponse 请求和响应对象作为参数。

destory()：在停止并且卸载 Servlet 时执行，负责释放资源

初始化阶段：Servlet 启动，会读取配置文件中的信息，构造指定的 Servlet 对象，创建 ServletConfig 对象，将 ServletConfig 作为参数来调用 init()方法。所以选 ACD。B 是在调用 service 方法时才构造的。

81.【多选题】【语言概念】以下分别对变量 a 给出定义，正确的有（）

A 一个有 10 个指针的数组，该指针指向同一个整型数:int *a[10];

B 一个指向 10 个整型数组的指针:int (*a)[10];

C 一个指向函数的指针，该函数有一个整型数并返回一个整型数:int *a(int);

D 一个有 10 个指针的数组，该指针指向一个函数，该函数有一个整型参数并返回一个整型数：int (*a[10])(int);

答案：ABD

解析：

C 改为：int (*a)(int)

指针数组：首先是一个数组，数组里面的元素都是指针；（存储指针的数组）

数组指针：首先是一个指针，指针指向一个一维数组；（指向数组的指针）

函数指针：一定要理解，回调中经常使用函数指针；

指针函数：就是一个普通函数，只是返回值是指针形式；

82.【多选题】【语言概念】下列排序算法中最好情况和最坏情况的时间复杂度相同的是？

A 堆排序

B 快速排序

C 冒泡排序

D 归并排序

答案：A C D

解析：

堆排序在最好和最坏情况下的时间复杂度均为 O(nlogn)

快速排序最好和最坏情况下的时间复杂度分别为 O(nlogn)和 O(n^2)

冒泡排序排序在最好和最坏情况下的时间复杂度均为 O(nlogn)

归并排序在最好和最坏情况下的时间复杂度均为 O(nlogn)

83.【填空题】【语言概念】定义 int**a[3][4]，则变量占有的内存空间为_____

96（64 位）或 48

32 位机器是 48（重点寻址字长）

12 个指针 每个指针 4 12*496

84.【简答题】【语言概念】解释局部变量、全局变量和静态变量的含义。

局部变量：有效使用范围在创建他的函数内部。

全局变量：全局变量既可以通过某对象函数创建，也可以是在本程序任何地方创建的。其作用域是整个源程序，可以被本程序所有对象或函数引用。

静态变量：在 C 语言中，static 关键字不仅可以用来修饰变量，还可以用来修饰函数。在使用 static 关键字修饰变量时，我们称此变量为静态变量。

85.【简答题】【语言概念】说明 fopen 函数中以下模式的区别 "r" "r+" "w" "w+" "a" "a+"

r：可读，不可写，必须存在，可在任意位置读取，文件指针自由移动

w：不可读，可写，可以不存在，若存在则必会擦掉原有内容从头写，文件指针无效

a：不可读，可写，可以不存在，必不能修改原有内容，只能在结尾追加写，文件指针无效

r+：可读可写，必须存在，可在任意位置读写，读与写共用同一个指针

w+：可读可写，可以不存在，必会擦掉原有内容从头写，文件指针只对读有效 (写操作会将文件指针移动到文件尾)

a+：可读可写，可以不存在，必不能修改原有内容，只能在结尾追加写，文件指针只对读有效 (写操作会将文件指针移动到文件尾)

86.【简答题】【语言概念】局部变量能否和全局变量重名？

不能，能，局部会屏蔽全局。

87.【简答题】【语言概念】队列和栈有什么区别？

队列先进先出，栈后进先出。

88.【简答题】【语言概念】如何引用一个已经定义过的全局变量?

1. 用 extern 关键字方式

2. 用引用头文件的方式

89.【简答题】【语言概念】全局变量可不可以定义在可被多个.c 文件包含的头文件中?为什么?

可以，在.h 文件中 声明该全局变量 extern int data; 在其它需要使用的源文件中 extern int data; 即可，但是该变量的定义只能有一份，只允许在一个源文件中定义该变量 int data = 1;

90.【简答题】【语言概念】计算 int *p 占用的内存大小

sizeof(int *p)

表示计算指向整型的指针变量 p 所占的字节数。

91.【简答题】【语言概念】 如何在程序中实现死循环

for（;;） while（1）

92.【简答题】【语言概念】do…….while 和 while.….do 有什么区别?

do while 条件不成立都一定可以执行一次。

93.【简答题】【语言概念】预处理器标识#error 的目的是什么?

#error //用于在编译阶段抛出错误信息，适合检查程序员在预处理阶段的限制不一致或者违规行为。

94.【简答题】【语言概念】关键字 static 的作用是什么?

static 可以修饰变量、方法、代码块和内部类

static 变量是这个类所有，由该类创建的所有对象共享同一个 static 属性

可以通过创建的对象名.属性名 和 类名.属性名两种方式访问

static 变量在内存中只有一份

static 修饰的变量只能是类的成员变量

static 方法可以通过对象名.方法名和类名.方法名两种方式来访问

static 代码块在类被第一次加载时执行静态代码块，且只被执行一次，主要作用是实现 static 属性的初始化

static 内部类属于整个外部类，而不属于外部类的每个对象，只可以访问外部类的静态变量和方法

95.【简答题】【语言概念】用预处理指令#define 声明一个常数，用以表明 1 年中有多少秒（忽略闰年问题）

#define SECONDS_PER_YEAR (60 * 60 * 24 * 365)UL

1、#define 语法的基本知识（例如：不能以分号结束，括号的使用，等等）

2、懂得预处理器将为你计算常数表达式的值，因此，直接写出你是如何计算一年中有多少秒而不是计算出实际的值，是更清晰而没有代价的。

3、意识到这个表达式将使一个 16 位机的整型数溢出-因此要用到长整型符号 L,告诉编译器这个常数是的长整型数。

4、如果你在你的表达式中用到 UL（表示无符号长整型），那么你有了一个好的起点。记住，第一印象很重要。

96.【简答题】【语言概念】关键字 const 是什么含意?

（1）欲阻止一个变量被改变，可以使用 const 关键字。在定义该 const 变量时，通常需要对它进行初始化，因为以后就没有机会再去改变它了；

（2）对指针来说，可以指定指针本身为 const，也可以指定指针所指的数据为 const，或二者同时指定为 const；

（3）在一个函数声明中，const 可以修饰形参，表明它是一个输入参数，在函数内部不能改变其值；

（4）对于类的成员函数，若指定其为 const 类型，则表明其是一个常函数，不能修改类的成员变量。

97.【简答题】【语言概念】关键字 volatile 有什么含意 并给出三个不同的例子。

volatile 是指易改变的，用他修饰的变量表明该变量是易发生改变的变量，每当优化器访问该变量时，都会重新读取该变量的值，而不是直接去找寄存器中找该变量的备份。

例子：1、并发的硬件寄存器，如状态寄存器。

2、中断服务器的子程序访问的非自动变量。

3、多线程中被多个任务共享的变量。

98.【简答题】【语言概念】Typedef 在 C 语言中频繁用以声明一个已经存在的数据类型的同义字。也可以用预处理 器做类似的事。

例如，思考一下下面的例子：#define dPS struct s *typedef struct s* tPS;

以上两种情况的意图都是要定义 dPS 和 tPS 作为一个指向结构 s 指针。哪种方法更好呢？（如果有的话）为什么？

前者是宏声明，此后 dPS 等价于后面的 struct s*；

后者是定义，此后 tPS 作为一个指针可以被使用；

99.【简答题】【语言概念】C 语言同意一些令人震惊的结构，下面的结构是合法的吗，如果是它做些什么？

```
int a= 5,b=7,c;
c=a++++++b;
(a++)+(++b)
```

a 原值代入，在该表达式处理完毕后，再自增处理

b 先自增处理，再将自增后的值代入该表达式

100.【简答题】【语言概念】关键字 static 的作用是什么？

static 可以修饰变量、方法、代码块和内部类

static 变量是这个类所有，由该类创建的所有对象共享同一个 static 属性

可以通过创建的对象名.属性名 和 类名.属性名两种方式访问

static 变量在内存中只有一份

static 修饰的变量只能是类的成员变量

static 方法可以通过对象名.方法名和类名.方法名两种方式来访问

static 代码块在类被第一次加载时执行静态代码块，且只被执行一次，主要作用是实现 static 属性的初始化

static 内部类属于整个外部类，而不属于外部类的每个对象，只可以访问外部类的静态变量和方法

101.【单选题】【语言基础】下面程序的输出是（）

```
Void main ()
{
Char a;
Char *str = &a;
Strcpy(str, “hello”);
Printf(str);
}
```

答案：无输出

解析: 没有为 str 分配内存空间，将会发生异常问题出在将一个字符串复制进一个字符变量指针所指地址。虽然可以正确输出结果，但因为越界进行内在读写而导致程序崩溃。

102.【单选题】【语言基础】以下程序输出结果是（）

```
void fun(int x, int y, int z)
{z = x*x+y*y; }
void main()
{
int a = 31;
    fun(5, 2, a);
Printf(“%d”, a);
}
```

A、0　　B、 29　　C、 31　　D、 无定值

C

解析:函数 fun 的形参是简单变量，main 函数中调用 fun 时只是把实参的值传递给形参，形参的改变不影响实参，所以调用完函数 fun 后 a 的值不发生变化，即 a=31。

103.【单选题】【语言基础】已知 int x = 10, y = 20, z = 30; 以下语句执行后 x, y, z 的值是（）。

```
If(x > y)
z = x; x = y; y = z;
```

A、x = 10, y = 20, z = 30　　B、 x = 20, y = 30, z = 30

C . x = 20, y = 30, z = 10　　D、 x = 20, y = 30, z = 20

答案：B

解析：对于 if 判断语句来说，如果 if()里面只有一个语句，则不用写括号。所以题目的主要意思为:

```
if(x > y)
{
  z = x;
}
```

```
x = y;
y = z;
```

所以，将 x，y，z 的值带入计算即可。

104.【单选题】【语言基础】下面程序输出是（）

```
void main()
{
Char *s = “12134211”;
Int v1 = 0, v2 = 0, v3 = 0, v4 = 0, k;
For(k = 0; s[k]; k++) {
        switch(s[k]) {
Default:   v4++;
Case ‘1’:  v1++;  break;
Case ‘3’:  v3++;
Case ‘2’:  v2++;  break;
}
}
Printf(“v1=%d, v2=%d, v3=%d,v4= %d\n”, v1, v2, v3, v4);
}
```

A、V1=4, v2=2, v3=1, v4=1　　**B、 V1=5, v2=3, v3=1, v4=1**

B、V1=5, v2=8, v3=6, v4=1　　**D、 V1=8, v2=8, v3=8, v4=8**

答案：B

解析：

这个程序原本的意思应该是找出字符串 s 中字符‘1’‘2’‘3’‘4’出现的个数，但是，标准的 switch case 语句应该是每个 case 后对应会有一个 break；如果没有 break，会继续往下执行。

第一个字符‘1’时，v1++；第二个字符‘2’时，v2++；第三个字符‘1’时，v1++；第四个字符‘3’时，v3++，v2++；第五个字符‘4’时，v4++，v1++；第六个字符‘2’时，v2++；第七个字符‘1’时，v1++；第八个字符‘1’时，v1++；

105.【单选题】【语言基础】有如下说明：int a[10] = {1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10}， *p = a; 则数值为 9 的表达式是（）

A、*P+9　　B、 *(p+8)　　C、 *p+ = 9　　D、 p+8

答案：B

解析：记住这样一个无条件相等的等式：*(p+i)与 P[i]无条件等价。因为 p 是指向数组 a 首元素的地址，所以 p[i]又和 a[i]等价。易知 B 选项即为 p[8]，也即为 a[8]，其值为 9。

106.【单选题】【语言基础】假定一个二维数组的定义语句为“int a[3][4] = { {3,4}, {2,8,6}};”，则元素 a[1][2]的值为（）

A、6　B、4　C、2　D、8

答案：A

解析：int a[3][4]表示定义一个 3 行 4 列的整型数组。最外围{}中的每一个{}表示依次对每一行赋值，不够的补 0}，所以{{3,4}，{2,8,6}}表示对数组的前两行赋值，每行也是依次赋值，不足补 0。

如图：

0　1　2　3
0　3　4　0　0
1　2　8　6　0
2　0　0　0　0

所以，a[1][2] = 6

107.【单选题】【语言基础】为了向二进制尾部增加数据，打开文件的方式应采用（A）

A、"ab"　　B、"rb+"　　C、 "wb"　　D、"wb+"

108.【单选题】【语言基础】执行语句"k=7>>1;"后，变量 k 的当前值是（C）

A、15　　B、31　　C、3　　D、1

109.【单选题】【语言基础】如变量已经正确定义，表达式(j=3,j++)的值是（A）

A、3　　B、4　　C、5　　D、0

110.【单选题】【语言基础】线性表(a1,a2,…,an)以链接方式存储时,访问第 i 位置元素的时间复杂性为(C)

A O(i)　　B O(1)　　C O(n)　　D O(i-1)

111.【单选题】【语言基础】有一个如下的结构体：

```
struct A{
long a1;
short a2;
int a3;
int *a4;
};
```

112.【单选题】【语言基础】请问在 64 位编译器下用 sizeof(struct A)计算出的大小是多少?（A）

A 24
B 28
C 16
D 18

113.【单选题】【语言基础】对一个含有 20 个元素的有序数组做二分查找，数组起始下标为 1，则查找 A[2]的比较序列的下标为(B)

A 9,5,4,2
B 10,5,3,2
C 9,6,2
D 20,10,5,3,2

114.【单选题】【语言基础】下列 Java 函数的执行结果是什么（）

```
static boolean foo(charc)
{
    System.out.print(c);
```

```
    return true;
}
public static void main(string[] args){
    int i = 0;
    for(foo('B');foo('A')&&(i<2);foo('C'))
     {
            i++;
         foo('D');
     }
}
```

A BADCBDCB

B BACDBACD

C BADCADCA

D 运行时抛出异常

答案：C

解析：

1.其实 foo('B');就是初始化条件，只会执行一次，所以第一个打印的肯定是 B。

2.因为 i=0;循环条件是 i<2 （由此可知，循环 i 等于 2 的时候就会停止循环)，所以 0<2 满足条件，接着会输出 A。然后执行 i++;i 就变成 1 了，在输出 D ，在最后输出 C。一次循环后的结果是：BADC。

3.第二次循环的开始是 foo('B');是初始条件，所以不会执行。直接从 foo('A')开始，输出 A，然后 i 为 1，且小于 2，此时循环体内再次执行 i++；i 的值为 2 了，再次输出 D，最后输出 C。第二次循环输出：ADC。

4.然后循环再次执行 for(foo('B');foo('A')&&(i<2);foo('C'))，直接输出 A。i 的值在第二轮循环后的值变成了 2，2<2 不成立，终止循环，输出 A。

故输出结果是：BADCADCA。

115.【单选题】【语言基础】若有宏定义:#define MOD(x, y) x%y,则执行一下语句之后的输出结果是（A）

```
        int a=13, b=94;
    printf("%d\n", MOD(b, a+4));
```

A、5　　B、7　　C、8　　D、11

116.【单选题】【语言基础】下列程序段的时间复杂度是（）

```
int fact(int n){
    if(n<=1){
        return 1;
    }
    return n*fact(n-1);
}
```

A O(log2n)

B O(nlog2n)

C O(n)

D O(n*n)

答案：C

解析：

当 n<=1 时执行 return 1 这一个语句，每次返回上一层都执行 n*fact(n-1)这一个语句，共执行 n-1 次。因此共执行基本语句 n 次，时间复杂度为 O(n)

117.【单选题】【语言基础】.若已知一个栈的入栈顺序是 1,2,3...,n,其输出序列为 P1,P2,P3,....Pn,若 P1 是 n,则 Pi=（）？

A i

B n-i+1

C 不确定

D n-i

答案：B

解析：

栈的排列遵循先进后（即后进先出）出的原则

因为 P1 是 n，是出栈的第一个数字，说明在 n 之前进栈的数字都没有出栈。所以这个顺序是确定的。

还可以知道，最后出栈的一定是数字 1，也就是 Pn。代入这个式子，是正确的。

118.【单选题】【语言基础】在 MySql 中， productname regexp '[1-3]xiaomi'的含义是（D）

A productname 匹配“xiaomi 重复 1 次或 5 次”的字符串
B productname 匹配“xiaomi 字符串前一个字符为 1 或 3“的字符串
C productname 匹配“xiaomi 重复 1 到 3 次”的字符串
D productname 匹配“xiaomi 字符串前一个字符为 1 到 3“的字符串

84.【单选题】【语言概念】同个进程的不同线程以下不能被共享的是？（）

A 全局变量
B 堆
C 文件句柄
D 栈

答案：B

解析：

线程共享的进程环境包括：

进程代码段、进程的公有资源（如全局变量，利用这些共享的数据，线程很容易的实现相互之间的通信）、进程打开的文件描述符、消息队列、信号的处理器、进程的当前目录、进程用户 ID、进程组 ID

线程独占资源：

线程 ID、寄存器组的值、用户栈、内核栈（在一个进程的线程共享堆区（heap））、错误返回码、线程的信号屏蔽码、线程的优先级

119.【单选题】【语言基础】某二叉树的中序遍历序列为 32145，后序遍历序列为 32145，则前序遍历序列为

A 54123
B 32154
C 32541
D 54321

答案：A

解析：

二叉树的中序遍历序列为 32145 ，后序遍历序列为 32145 ，可知该树只有左子树结点，没有右子树结点， 5 为根结点。

中序遍历序列与后序遍历序列相同，说明该树只有左子树没有右子树，因此该树有 5 层，从顶向下依次为

54123 。

120.【单选题】【语言基础】一下程序的返回值是多少（）

```
#define product(x)(x*x)
int main()
{
int i= 3,j,k;
j= product(i++);
k = product(++i);
printf("%d %d",j,k);
}
```

A、9 25

B、16 25

C、12 30

D、12 49

答案：A

解析： j= product(i++)是先将 i 的值带入后 i 才自增。

121.【单选题】【语言基础】以下程序输出结果是什么（ ）

```
int main()
{
int arr[]={11,12,13,14,15};
int *ptr = arr;
*(++ptr)+= 100;
printf("%d %d\n",*(ptr),*(ptr++));
return 0;
}
```

A、11 112

B、13 112

C、11 12

D、111 13

答案：B

解析：printf 是从 *(ptr++)开始处理的的，加上前面的一次自增所以 ptr 自增了两次。*(++ptr)是 ptr 自增完然后再取它所指的那个数进行计算。

122.【单选题】语言基础】C 语言中的标识符只能由字母、数字和下划线三种字符组成，且第一个字符（C）

A、必须为字母

B、必须为下划线

C、必须为字母或下划线

D、可以是字母，数字和下划线中任一种字符

123.【单选题】【语言基础】 以下代码输出值是多少（）

```
int i,j,k,m;
i=(j=4,k=8,m=16);
printf("%d",i);
```

A、1

B、4

C、8

D、16

答案：D

解析：覆盖导致 i 最后等于 16

例如：int i=(j=4,k=8,l=16,m=32);

则等同于：

int j=4, k=8, l=16, m=32;

int i = j;

int i = k;

int i = l;

int i = m;

z 最后 i = m =32，故输出为 32

124.【单选题】【语言基础】a=5;b=6; a+=b++;printf（"a=‰d，b=%d"）;最终结果是多少（）

a=11,b=6

a=12,b=6

a=11,b=7

D、a=12,b=7

答案：C

解析：因为 a+=b++ 是先算 a+b 然后 b 再自增。

125.【单选题】【语言基础】int a[10];问下面哪些不可以表示 a[1]的地址（）

A、a+sizeof(int)

B、&a[0] +1

C、(int*)&a+1

D、(int*)((char*)&a+sizeof(int))

答案：A

解析：&a[0]+1 与 a+1 和&a[1]是等价的是等价的，但 a+sizeof(int) 是 a+4 错。

126.【单选题】【语言基础】以下程序的运行结果是（B）

```
Int main(void)
{
   Int i=8;
   If(++i>8)
  {
Printf(“%d\n”,i--);

}
Else
{ printf(“%d\n”,i++);
}
}
```

A、7 B、8 C、9 D、10

127.【单选题】【语言基础】在 c 语言中，要求运算数必须是整形的运算符是（D）

A、/　　B、++　　C !=　　D、%

128.【单选题】【语言基础】在 32 位系统下，请计算以下 sizeof 的值（B）

```
Char str[]=”helloword”;
Char array[10]=”helloword”;
Char *p=”helloword”;
Sizeof(str)=();sizeof(array)=();sizeof(p)=()
```

A、4　　B、10　　C、11　　D、12

129.【单选题】【语言基础】有如下程序，输出是（B）：

```
Int main(void)
{
    Int s=1;
    Int a=0,b=0;
    Switch(s)
{
   Case 0:b++;
   Case 1:a++;
   Case 2:a++;b++;
}

Printf(“a=%d,b=%d\n”,a,b);
Return 0;
}
```

A、a=2,b=1 B、a=1,b=1　C、a=1,b=0　D、 a=2,b=2

130.【单选题】【语言基础】以下程序的运行结果是（A）

```
Void swap(int a,int b)
{
```

```
   Int tmp;
   tmp=a;
a=b;
b=tmp;
}
Int main(void)
{
   Int a=10;
Int b=20;
Swap(a,b);
Printf(%d,%d\n,a,b)
Return 0;
}
```

A、10,20　　B、20,10　　C、10,10　　D、20,20

131.【单选题】【语言基础】数组定义为 inta[10];则下列表达式错误的是（D）

A、A*a　　B、&a[0]　　C、a　　D、a++

132.【单选题】【语言基础】以下程序的执行结果是（D）

```
#define sum(x,y) x+y
Int main(void)
{
   Int a=2,b=5;
Int c=0;
C=sum(a,b)*10;
Return 0;
}
```

A、20　　B、25　　C、52　　D、70

133.【单选题】【语言基础】有如下程序，请问 for 将循环多少次（B）

```
Int main(void)
```

```
{
  Unsigned int i;
For(i=10,i>=0;--i)
{
  ---
}\return0
}
```

A 9　B、10　　C、11　　D、无数次

134.【单选题】【语言基础】在 16 位机器上跑下列 foo 函数的结果是（）

```
Void foo( )
{
int i = 65536;
cout << i <<”,”;
I = 65535;
cout << i;
}
```

A、-1，65535　　B、0，-1　　C、-1，-1　　D、0，65535

答案：B

解析：在 16 位机器上, 65536 值保留较小的 16 位, 全 0;65535 的 16 位全 1, 为负数, 其绝对值的计算方式是 - 1 求法, 得 -1.

135.【单选题】【语言基础】64 位系统上，定义变量 int *a[2][3]占据（）字节

A、4　B、12　C、24　D、48

答案：C

解析：int*a[2][3] a 的类型是 int[][],实际存储的是 a[0]的地址，占 8 字节，同时创建出两个 int[]数组，两个 int[]数组实际存储的分别是 a[0][0]和 a[1][0]的地址，一共占 16 字节。每个 int 占 8 字节，所以选 C、

注意，这里的定义是 int*，也就是说，只是定义了数组，并没有对该数组赋值。也就是说，int[]数组里面存储的地址是并没有初始化，且 int 没有被创建。所以是 24

136.【单选题】【语言基础】当 n = 5 时，下列函数的返回值是：（）

```
int   foo(int n)
{
If(n < 2)
{
return n;
}
else
return foo(n - 1) + foo(n - 2);
}
```

A、5　B、7　C、8　D、10

答案：A

解析：foo(5) = foo(4) + foo(3)，foo(4) = foo(3) + foo(2)，foo(3) = foo(2) + foo(1)

foo(2) = foo(1) + foo(0)

又由 n<2 return n 可知 foo(1) = 1 foo(0) = 0

所以 foo(2) = 1 + 0 = 1，foo(3) = 1 + 1 = 2，foo(4) = 2 + 1 = 3，foo(5) = 3 + 2 = 5

137.【单选题】【语言基础】下列程序的输出是（）

```
#define add(a，b)    a+b
Int main( )
{
printf(“ %d\n”, 5*add(3,4));
return 0;
}
```

A、23　B、35　C、16　D、19

答案：D

解析：本题使用了宏定义，在代码编译时，先进行宏定义替换，即 printf("%d\n",5*add(3，4)) = printf("%d\n",5*3 + 4)，5 * 4 + 4 = 19，因此本题正确答案是 D

138.【单选题】【语言基础】补充下面函数代码（）

如果两端内存重叠，用 memcpy 函数可能会导致行为未定义。而 memmove 函数能够避免这样问题，下面是一种实现方式，请补充代码。

A、pStr1 < pStr2　　str1

B、pStr1+n < pStr2　str2

C、pStr1+n < pStr2　||　pStr2+n < pStr1　　str2

D、pStr2+n < pStr1　str1

答案：A

解析：目标地址小于源地址时，可以从前向后拷贝

139.【单选题】【语言基础】输出结果为（）

```
int func(int a)
{
int b;
switch(a)
{
case 1: b = 30;
case 2: b = 20;
case 3: b = 16;
default: b = 0;
}
return b;
}
```

则 func(1) = ?

A、30　　B、20　　C、16　　D、0

答案：D

解析：如果 case 之后没有 break;就会顺序执行一直到最后的 default

140.【单选题】【语言基础】C 语言中，下列运算符优先级最高的是（A）

A、！　　B、%　　C、>>　　D、==

141.【单选题】【语言基础】能正确表示 “当 x 的取值在[1, 10] 和 [200, 210]范围内为真，否则为假” 的表达式是 ( )

A、(x > 1)&&(x <= 10) && (x >= 200) && (x <= 210)

B、(x > 1)||(x <= 10) || (x >= 200) || (x <= 210)

C、(x > 1)&&(x <= 10) || (x >= 200) && (x <= 210)

D、(x > 1)||(x <= 10) && (x >= 200) || (x <= 210)

答案：C

解析: C 语言中，&&左右两边表达式同时为真才为真，||左右两边表达式一个为真就为真。所以，对于 A 选项来说，需要四个不等式同时成立才为真。但是无法找到满足这个条件的数，所以 A 错误。对于 B 来说，只有四个不等式满足任意一个就为真。当 x 为-1 时，满足 x<=10，x<=210 两个条件，但题目想要找出 1 到 10,200 到 210 两个区间内的值，所以 B 错误。对于 D 来说，&&左右两边表达式同时满足才为真。还是以-1 为例，左边满足 x<=10，右边满足 x<=210，但与提议不符，所以选 C。

142.【单选题】【语言基础】下面程序段的运行结果是（）

```
Int x = y = 0;
While(x < 15)   y++, x+=++y;
Printf(“%d, %d”, y, x);
```

A、20, 7　　B、 6, 12　　C、 20, 8　　D、 8, 20

答案：D

解析:

首先，要弄清 y++与++y 的区别。

++y 表示先给 y 加上 1，再进行其它运算;

y++表示先让 y 进行其它运算，再给 y 加上 1。

例如:

```
#include <stdio.h>
int main()
{
int y = 1;
int x = 1;
int a = y++;
int b = ++x;
```

```
printf("%d,%d", a, b);
return 0;
}
```

上述代码的输出结果为 1,2。对于 int a = y++;来说，因为是 y++，所以先让 y 进行赋值的运算，即把 y = 1 赋值给 a，再给 y 加上 1;对于 int b = ++x;来说，因为是++x，所以先让 x 加上 1，再把加 1 后的 x 赋值给 b。所以 a 为 1，b 为 2。

再来看题目的代码，以第一次循环为例。第一次进入 while 循环时，x 和 y 都是 0。先执行 y++，此时对于 y++来说，并无其他多余的运算，所以直接给 y 加上 1(此时 y = 1)。再来看 x += ++y;因为是++y,所以先给 y 加上 1 再进行 x += y 运算(此时 y=2，x = 2),。所以，第一次结束 while 循环时，x = 2，y = 2。以此类推，最后可以算出 D 答案。

143.【单选题】【语言基础】对两个数组 a 和 b 进行如下初始化 char a[ ] = “ABCDEF”; char b[ ] = {‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’};则以下叙述正确的是（）

A、a 与 b 数组完全相同　　B、 a 与 b 长度相同

B、a 和 b 中都存放字符串　　D、 a 数组比 b 数组长度长

答案：D

解析：对于 a 数组来说，是一个字符串数组，所以数组的最后一位是’\0’但不会显示出来。而 b 则是一个字符数组，长度就是数组内字符的个数。所以答案选 D

144.【单选题】【语言基础】以下这 3 个函数哪一个最可能引起指针方面的问题（）

```
Int *f1 ( void )
{
Int x = 10;
Return ( &x );
}
Int *f2( void )
{
Int *ptr;
*ptr = 10;
Return ptr;
}
Int *f3( void )
```

```
{
Int *ptr;
Ptr = (int *) mallloc (sizeof( int ));
Return ptr;
}
```

A、只有 f3　　B、 只有 f1 and f3　　C、 只有 f1 and f2　　D、 f1 f2 f3

答案：C

解析:

f1 显然有问题，它返回一个局部变量的指针，局部变量是保存在 stack 中的,退出函数后，局部变量就销毁了，保留其指针没有意义，因为其指向的 stack 空间可能被其他变量覆盖了

f2 也有问题,ptr 是局部变量，未初始化，它的值是未知的，*ptr 不知道指向哪里了，直接给*ptr 赋值可能会覆盖重要的系统变量，这就是通常说的野指针的一种

145.【单选题】【语言基础】以下不是无限循环的语句为（）

A、for(y = 0, x = 1; x > ++y; x=i++)　i=x;　　B、 for(;; x++ = i);

C、While(1){x++;}　　D、 for(i = 10; ; i--) sum+=i;

答案：A

解析: 对于 A 来说，第一次 for 循环判断时 x 和 y 都是 1，不会进入 for 循环内部。

146.【单选题】【语言基础】若 i 为整型变量， 执行语句 for(i = 1; i++<4;);后变量 i 的值是（）

A、3　　B、 4　　C、 5　　D、 不定

答案：C

解析: 将 i=1 带入 for 循环中就可得到答案。

147.【单选题】【语言基础】下面程序的输出是（）

```
Void main( )
{
Unsigned char ucNum;
For (ucNum = 0; ucNum < 500; ucNum++){
```

```
.....
}
Printf(“%d”, ucNum);
}
```

A、499　　B、 500　　C、 501　　D、 无输出

答案：D

解析: 死循环,unsignedint 的取值范围是 0~255

148.【单选题】【语言基础】下面这段代码的输出结果为（A）：

```
#include
void change(int*a, int&b, int c)
{
c=*a;
b=30;
*a=20;
}
int main ( )
{
int a=10, b=20, c=30;
change(&a,b,c);
printf(“%d,%d,%d,”,a,b,c);
return 0;
}
```

A 20，30，30
B 10，20，30
C 20，30，10
D 10，30，30

149.【单选题】【语言基础】下面函数的功能是（）

```
Int fun(char *s)
{
Char *p = s;
```

```
While(*p++);
Return p-s-1;
}
```

A、计算字符串的位（bit）数　　B、复制一个字符串

B、求字符串的长度　　D、求字符串存放的位置

答案：B

解析：While(*p++);此句执行结束时，p 指向的是字符串末尾的"\0"，此时 s 仍然是指向字符串的开头所以 p-s-1 是求字符串的长度，-1 就是去掉"\0"的长度 1

150.【单选题】【语言基础】8 进制数 256，转化为 7 进制数是（）

A、356　B、336　C、338　D、346

答案：B

解析：八进制 256 转化成十进制 174，然后转换成七进制 336

151.【多选题】【语言基础】关于红黑树和 AVL 树，以下哪种说法正确?（ABC）

A 两者都属于自平衡二叉树

B 两者查找,插入，删除的时间复杂度相同

C 包含 n 个内部节点的红黑树的高度是 O(log(n))

D JDK 的 TreeMap 是一个 AVL 的实现

152.【填空题】【语言基础】完成代码填空

```
int func(int a)
{
int b=O;
switch(a)
{
case 1:b=30;
case 2:b= 20;
case 3:b=60;
default: b=70;
}
```

```
return b;
}
```
则运行函数 func（2）= ____

70

default 就是最终不满足才输出的.

153.【填空题】【语言基础】完成代码填空

```
int a[3];
a[0]=0;a[1]=1;a[2]=2;
int *p,*q;
p=a;
q=&a[2];则 a[q-p]=_____
```

2 p 是首地址　q 是 a【2】的地址

154.【简答题】【语言基础】请列出你知道的常用的开发工具，要求编辑器，编译器，调试器，版本管理工具至少各有一个。

1.开发 iOS 系统的工具——xcode 软件开发；

2.开发 Android 系统的工具——eclipse 软件开发；

3.android app 界面设计的软件工具——App UI Designer；

4.ios app 应用界面设计软件工具——ProtoShare；

5.测试软件的工具—— Emacs、vim、Notepad++（Windows）、UltraEdit（Windows）、TextPad（Windows）；

6.开发 web APP 的工具——Editplus 开发工具、UltraEdit 手机网站工具；

7.提升 web APP 开发效率的工具——Google Web Designer（无需懂得 html5 语言）、Gauge.js（自定义动画仪表和滑动杆）、Timesheet.js、Quintus（开发 web 游戏 app 软件的有利工具）、NoMe；

8.简化 ios 开发的工具——Kinvey 苹果 app 工具、Firebase IOS 开发工具、IOS Boilerplate 工具软件、Slash 移动 app 开发工具；

9.建设商城网站响应式设计工具——响应式线框图、Wirefy 设计工具、MockUphne 原型设计工具；

10.傻瓜式开发工具——AppMakr、App Press、Apepery、GoodBarber、Appmachine、iBuildApp~

155.【简答题】【语言基础】C 语言变量定义语言中 x 的数据类型是什么?y 执行后的结果是什么？

```
int *x:
int y,z=2;
x=&z;
y=*x;
```

x 的数据类型是 int 型指针；y 执行后的结果是 2

156.【简答题】【语言基础】Using C check if the file exists (List all methods you know)

方法一：access 函数判断文件夹或者文件是否存在

方法二：fopen 函数判断文件是否存在

157.【简答题】【语言基础】C 语言中用 inline 修饰函数定义是什么含义？在什么场合应用？有什么利弊？

定义为 inline 函数之后，会省去函数调用的开销，直接嵌套汇编代码，取代函数调用，提高效率。当我们过度使用 inline 函数，会造成程序文件变大，性能降低。

158.【简答题】【语言基础】用一行 C 代码实现将 unsigned long register 的第 0,4,6 位清 0？

32 位下：register&=0xffffffae

分析：0,4,6 清 0；10101110 对应着：1010 对应 a; 1110 对应 e

159.【简答题】【语言基础】写一个函数，把给定整数的二进制表示形式中“1”的个数统计出来。

```
#include<iostream>
using namespace std;

int CoutOne(int a){
    int num=0,b;
```

```
    do{
        b=a&1;                          //取出二进制末尾的数字
        if(b==1) num++;
        a>>=1;                          //将形参右移一位
    }while(a!=0);                       //当形参为 0 时，结束循环

    return num;
}

int main(){
    int a;
    cin>>a;
    cout<<CoutOne(a)<<endl;
    return 0;
}
```

160.【编程题】【语言基础】编写一个函数，要求输入任意两个无符号 32 位整形数，在控制台打印两个数相加的结果，要求禁止使用 64 位变量。

```
#include <setjmp.h>
#include <stdliB、h>
#include <stdio.h>
static jmp_buf buf;
void main()
{
    volatile int b;
    b=3;
    if(setjmp(buf)!=0)
    {
        printf("%d ",b);
        exit(0);
    }
    b=5;
    longjmp(buf,1);
}
```

161.【编程题】【语言基础】编写 strcpy 函数

```
#include <stdio.h>
#include <stdliB、h>
#include <assert.h>

//链式访问
char * my_strcpy(char *dest, const char *src)
{//将源字符串加 const，表明其为输入参数,起到相应的保护作用
    assert(src != NULL&&dest != NULL);//对源地址和目的地址加非 0 断言
    char *ret = dest;
    while ((*dest++ = *src++))
            ;
     return ret;//引用返回地址，方便链式操作!!
}

int main()
{
      char *p = "bit-tech";
      char arr[20];
      //strcpy(arr,p);
      printf("%s\n", my_strcpy(arr, p));
      system("pause");
      return 0;
}
```

162.【简答题】【语言基础】有数组定义 int a[2][2]={{1}，{2，3}};则 a[0][1]的值是否为 0？为什么？

是，程序默认的。编译器会自动帮你把每一维不足的部分以 0 填充。

163.【代码题】【语言基础】请找出下面代码中的所以错误

```
 Void test1()
{
char string[10];
```

```
char* str1="0123456789";
strcpy(string, str1);
}
strcpy(string, str1);这个将 int 类型给了 char 类型。
```

164.【代码题】【语言基础】请找出下面代码中的所以错误

```
 void test2()
{
char string[O];
if(strlen(str1)<=10)
{
Str1[i]=’a’;
}
Strcpy(string,str1);
}
}
```

Strcpy(string,str1)错误大写

char string[O]字母待替了数字

Stir1，stir1 错误

165.【代码题】【语言基础】请找出下面代码中的所以错误

```
void test3(char* str1)
{
char string[10];
If(strlen(str1)<=10)
{
strcpy(string,str1);
}
}
```

if(strlen(str1) ＜= 10)应改为 if(strlen(str1) ＜ 10)，因为 strlen 的结果未统计’\0’所占用的 1 个字节

166.【编程题】【语言基础】编写一个函数，要求输入年月日时分秒，输出该年月日时分秒的下一秒。如输入 2004 年 12 月 31 日 23 时 59 分 59 秒，则输出 2005 年 1 月 1 日 0 时 0 分 0 秒

```
void ResetTheTime(int *year,int *month,int *date,int *hour,int *minute,int*second)
{
    int dayOfMonth[12]={31,28,31,30,31,30,31,31,30,31,30,31};
    if( *year < 0 || *month < 1 || *month > 12 ||
        *date < 1 || *date > 31 || *hour < 0 || *hour > 23 ||
        *minute < 0 ||*minute > 59|| *second <0 || *second >60 )
        return;

    if( *year%400 == 0 || *year%100 != 0 && *year%4 == 0 )
        dayOfMonth[1] = 29;
    ++(*second);
    if(*second >= 60)
    {
        *second = 0;
        *minute += 1;
        if(*minute >= 60)
        {
            *minute = 0;
            *hour += 1;
            if(*hour >= 24)
            {
                *hour = 0;
                *date += 1;
                if(*date > dayOfMonth[*month-1])
                {
                    *date = 1;
                    *month += 1;
                    if(*month > 12)
                    {
                        *month=1;*year += 1;
                    }
```

```
                    }
                }
            }
        }
        return;
}
```

167.【编程题】【语言基础】不使用库函数，编写函数 int strcmp（char*source， char *dest）相等返回 0，不等返回-1

```
#include <stdio.h>
  #include <stdliB、h>
     int strcmp(char *source, char *dest)
{
while(*source == *dest && *source != '\0' && *dest != '\0')
{
   source++;
   dest++;
}
if (*source =='\0' && *dest == '\0')
   return 0;
else
 return -1;
  int main()
{
char *str1 = "abcde";
char *str2 = "abcde";
printf("ret = %d", mystrcmp(str1, str2));

return 0;
}
```

168.【程序题】【语言基础】设计实现 int atoi（char* s）

```
int atoi(const char *ptr)
{int i=0;
int val,result=0;
if(*ptr=='+'||*ptr=='-'||(*ptr>='0'&&*ptr<='9'))
; /*执行空语句*/
else return 0;/*第一个字母不是+ - 或数字字符则返回 0*/
if(*ptr=='+'||*ptr=='-')
{ptr++;
if(*(ptr-1)=='-')
i=-1;
}
while(*ptr>='0'&&*ptr<='9')
result=10*result+*ptr++-'0';
return i?-result:result;/*若 i 的值非 0 说明值为复值*/
}
```

169.【程序题】【语言基础】设计函数实现对数组的升序排列 int array【】={45，56，76，234，1，34，23，2，3）;（数组可以是任意值）

答案不唯一：

```
#include<stdio.h>
#include<iostream>
using namespace std;
int main()
{
int c;
cin>>c;
cout<<endl;
int b[c] ;
for(int i=0;i<c;i++)
cin>>b[i];
int sort (int *a , int count);
sort (b,c);
```

```
system("pause");
}
int sort(int *a, int count)
{
int i,j,k;
for(i=0;i<count;i++)
{
for(j=0;j<count-i-1;j++)
{
if(*(a+j)>*(a+j+1))
{ k=*(a+j) ;
*(a+j)=*(a+j+1);
*(a+j+1)=k;
}
}
}
for(i=0;i<count;i++)
cout<<*(a+i)<<" ";
cout<<endl;
}
```

170.【程序题】【语言基础】编写一个递归函数，删除一个目录里的所有内容

```
#include<stdio.h>
#include<stdliB、h>
typedef struct node
{
char data;
struct node* next;
}Lnode;
Lnode *head=NULL;
Lnode *tail=NULL;
void create()
{
head = (Lnode*)malloc(sizeof(Lnode));
tail = head;
```

```
if(head!=NULL)
{
printf("please input data: ");
scanf(" %c",&head->data);
head->next=NULL;
}
}
void delFirst(Lnode *pre ,Lnode *ptr,char x) //递归删除第一个 = x
{
if(ptr==NULL)
return;
if(pre==NULL && ptr->data == x)
{
head=ptr->next;
ptr->next = NULL;
free(ptr);
ptr=NULL;
return;
}
if(pre!=NULL && ptr->data == x)
{
pre->next = ptr->next;
ptr->next = NULL;
free(ptr);
ptr = NULL;
return;
}
pre = ptr;
ptr = ptr->next;
delFirst(pre,ptr,x);
}
void delAll() //递归删除所有
{
Lnode * tmp=NULL;
if(head==NULL)
return;
```

```
if(head->next!=NULL)
{
tmp = head->next;
free(head);
head=tmp;
delAll();
}
else
{
free(head);
head=NULL;
}
}
void insert()
{
if(tail == NULL)
return;
tail->next = (Lnode*)malloc(sizeof(Lnode));
if(tail->next != NULL)
{
printf("please input data: ");
scanf(" %c",&(tail->next->data));
tail=tail->next;
tail->next=NULL;
}
}
void print()
{
Lnode* tmp =head;
if(head==NULL)
printf("no data\n");
while(tmp!=NULL)
{
printf("%c ",tmp->data);
tmp=tmp->next;
}
```

```
}
void lfree() //非递归删除
{
Lnode* tmp=NULL;
while(head!=NULL)
{
tmp=head;
head=head->next;
free(tmp);
}
}
void main()
{
int i=0;
create();
for(i=0; i<5; i++)
{
insert();
}
delFirst(NULL,head,'1');
print(head);
delAll();
print(head);
}
```

171.【简答题】【语言基础】写一个"标准"宏 MIN，这个宏输入两个参数并返回较小的一个

```
#define MIN(A,B)((A)<=(B)?(A):(B))
```

172.【简答题】【语言基础】嵌入式系统中经常要用到无限循环，你怎么样用 C 编写死循环呢?

for（;;）或者 while（）

173.【简答题】【语言基础】下面的代码输出是什么，为什么？

```
void foo(void) {
unsigned int a = 6;
int b = -20;
(a+b > 6) puts("> 6") : puts("<= 6")
}
```

unsigned 是无符号数。 b 是有符号数负数，无符号数和有符号数混合运算 要注意 你到底 想计算什么。默认，把有符号数内容 当成无符号数解释 计算。-20 就当 0xffffffec (4294967276).显然(a+b > 6) 成立。输出 > 6

174.【简答题】【语言基础】评价下面的代码片断：

```
unsigned int zero =0;
unsigned int compzero OxfFFF;
/1's complement of zero */
```

~是位运算符，是取反的意思,即二进制位 0 变 1,1 变 0；

unsigned int compzero = 0xFFFF;表示 1111 1111 1111 1111，对于 int 型不是 16 位的处理器来说，上面的代码是不正确的。

unsigned int compzero=~0;（以 16 位处理器来说明）表示将 0 的二进制位 0000000000000000 取反，变成 1111111111111111，对不是 16 位处理器的也是正确的

175.【简答题】【语言基础】32 位的 ARM Linux 上，指针 char *ptr[2]占的内存空间为多少字节，为什么？

指针是多少位只要看地址总线的位数就行了。*80386* 以后的机子都是 *32* 的数据总线。所以指针的位数就是 *4* 个字节了。

176.【简答题】【语言基础】请问以下代码有什么问题：

```
 int   main() {
char a;
char *str=&a; strcpy(str,"hello");
printf(str);
return 0; }
```

没有为 *str* 分配内存空间，将会发生异常问题出在将一个字符串复制进一个字符变量指针所指地址。虽然可以正确输出结果，但因为越界进行内在读写而导致程序崩溃。

177.【简答题】【语言基础】给出一个字符 aBcdEfgH，请写一个函数打印其中的小写字母；

```
#include<stdio.h>
#include<string.h>
void fun(char a[], int len)
{
   for(int i=0;i<len;i++)
   {
      if(a[i]>=97&&a[i]<=122)
      {
         printf("%c",a[i]);
         continue;
      }
      continue;
   }
}
int main()
{
       char a[] = { "aBcdEfgH" };   //定义数组 a 并赋值
       int len = strlen(a);   //求数组 a 的长度
       fun(a, len);
       return 0;
}
```

178.【编程题】【语言基础】不调用 C++/C 的字符串库函数，请编写函数 strcpy 已知 stepy 函数的原型是 char *strepy(char *strDest, const char *stSrc);其中 strDest 是目的字符串 strSrc 是源字符串

方法一:

```
char *strcpy(char *strDest,const char *strSrc)
{
```

```
    assert((strDest!=NULL)&&(strSrc!=NULL));
    char *address=strDest;
while((*strDest++=*strSrc++)!='\0')
{
NULL;
    }
    return address;
}
```

方法二:

```
char *strcpy(char *strDest,const char *strSrc)
{
    assert((strDest!=NULL)&&(strSrc!=NULL));
    char *address=strDest;
while((*strDest++=*strSrc++)!='\0')
{
NULL;
    }
    return address;
}
```

179.【简答题】【语言基础】请使用递归算法编写求 N 的阶乘函数。

```
#include<stdio.h>
int f(int n)/*递归函数*/
{
    int fac;
if (n < 0)
{
printf("n<0,data error!");
}
else
{
   if (n == 0 || n == 1)
{
fac = 1;
```

```
}
   else
{
fac = f(n - 1) * n;
}
}
      return fac;
}
int main()
{
      int n, y;
      printf("请输入一个整数：\n");
      scanf_s("%d", &n);
      y = f(n);
      printf("%d!=%d", n, y);
      return 0;
}
```

答案：A

解析：设使用的是 p 进制，则 15*4=112 等价于：(p + 5) * 4 = p^2 + p + 2 解出来 p=-3（舍去）和 p=6

180.【编程题】【语言基础】实现列表删除函数，删除第 i 个节点

```
Void table_pop(table_node*data,int i)
{
Table_node* temp =data;
Int number =0;
While(number<i-1)
{
       Temp = temp->next;
Number++;
}
Temp->next = temp->next->next;
}
```

181.【编程题】【语言基础】写一个函数找出一个整数数组中， 第二大的数。 自己设计函数的形式参数和返回值。

```
const int MINNUMBER = 0;
int find_second_max(int data[],int count)
{
	int maxnumber = data[0];
	int sec_max = MINNUMBER;
	for (int i = 1; i < count; i++)
	{
		if(data[i] > maxnumber)
		{
			sec_max = maxnumber;
			maxnumber = data[i];
		}
		else
		{
			if (data[i] > sec_max && data[i] < maxnumber)
			{
				sec_max = data[i];
			}
		}
	}
	printf("%d",sec_max);
	return sec_max;
}
```

182.【简答题】【语言基础】用变量 a 给出下面的定义

a)一个整型数

b）一个指向整型数的指针

c）一个指向指针的的指针，它指向的指针是指向一个整型数

d）一个有 10 个整型数的数组

e）一个有 10 个指针的数组，该指针是指向一个整型数的

F）一个指向有 10 个整型数数组的指针

g）一个指向函数的指针，该函数有一个整型参数并返回一个整型数

h）一个有 10 个指针的数组，该指针指向一个函数，该函数有一个整型参数并返回一个整型数

a) int a; // An integer

b) int *a; // A pointer to an integer

c) int **a; // A pointer to a pointer to an integer

d) int a[10]; // An array of 10 integers

e) int *a[10]; // An array of 10 pointers to integers

f) int (*a)[10]; // A pointer to an array of 10 integers

g) int (*a)(int); // A pointer to a function a that takes an integer argument and returns an integer

h) int (*a[10])(int); // An array of 10 pointers to functions that take an integer argument and return an integer

183.【编程题】【语言基础】试写一个函数，计算字符串中最大连续相同的字符个数。例如，若 s 为"aaabbb"则返回值为 4:若 s 为"abede",则返回值为 1. (20 分)

```
int max_same_char(chart *s)
{
    char char_temp=s[0];
    int   length_temp =1;
    int Max_length = 0;
    int number = 1;
    while (s[number]!=0)
    {
        /* code */
        if(char_temp == s[number])
        {
            length_temp++;
        }
        else
        {
            if(Max_length<length_temp)
            {
                Max_length = length_temp;
            }
            length_temp = 0;
```

```
            char_temp = s[number];
        }
        number++;
    }
    return Max_length;
```

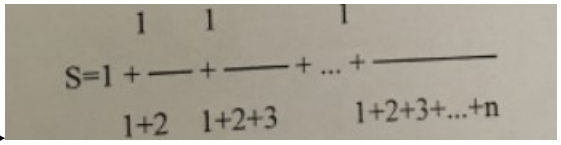
184.【编程题】【语言基础】试写一个函数检查表达式中的括号是否匹配,其中需要检查的括号包括()、[]、{}、<>，例如表达式中前面有’(‘但后面没有’)’则认为不匹配(20分）

```
#include<stdio.h>
#include<stdliB、h>
//栈
typedef struct
{
    char flage_char;
    int length;
}zhan;
zhan* zhan_init(char flage_data)
{
    zhan* new_temp = (zhan*)malloc(sizeof(zhan));
    new_temp->flage_char = flage_data;
    new_temp->length = 0;
    return new_temp;
}

//入栈
void zhan_into(zhan* data)
{
    data->length=data->length+1;
}
//出栈
void zhan_out(zhan *data)
{
```

```
    data->length=data->length-1;
}
int main ()
{
    //括号（匹配队列
    zhan* kuo_zhan = zhan_init('(');
    //方括号
    zhan* fang_zhan = zhan_init('[');
    //大括号
    zhan* da_zhan = zhan_init('{');
    //尖括号
    zhan* jian_zhan = zhan_init('<');

    //获得字符串

    char str[10000];
    scanf("%s",str);
    int number = 0;
    while (str[number]!='\0')
    {
        switch (str[number])
        {
            case '(':zhan_into(kuo_zhan);break;
            case '[':zhan_into(fang_zhan);break;
            case '{':zhan_into(da_zhan);break;
            case '<':zhan_into(jian_zhan);break;
            case ')':zhan_out(kuo_zhan);break;
            case ']':zhan_out(fang_zhan);break;
            case '}':zhan_out(da_zhan);break;
            case '>':zhan_out(jian_zhan);break;
        }
        number++;
    }
    if(da_zhan->length!=0||fang_zhan->length!=0||jian_zhan->length!=0||jian_zhan!=0)
    {
        printf("不匹配\n");
```

```
    }

    return 0;
}
```

185.【编程题】【语言基础】试写一个函数，函数的功能是:根据以下公式计算 s,计算结果作为函数值返回; n 通过参数传入



例如:若 n 的值为 11 时，函数的值为:1.83333

```
int getss(int i)
{
    int sum = 0;
    for (int j = 0; j <= i; j++)
    {
        sum += j;
    }
    return sum;
}

void function(int data)
{
    double temp=0;
    for (int i = 1; i <= data; i++)
    {
        temp += (double)1 / (double)(getss(i));
    }
    printf("%f", temp);
}
```

186.【简答题】【嵌入式系统】嵌入式系统总是要用户对变量或寄存器进行位损作。给定一个整型变量 a，写两段代码，第一个设置 a 的 bit 3，第二个清除 a 的 bit 3.在以上两个操作中，要保持其它位不变。

```
#define BIT3 (0x01<<3)
Static int a;
 Void set_bit3(void)
{
      a|=BIT3;
}
Void clear_bit3(void)
{
      a &= ~BIT3;
}
```

187.【简答题】【嵌入式系统】嵌入式系统经常具有要求程序员去访问某特定的内存位置的特点，在某工程中 ，要求 设置一绝对地址为 0x67a9 的整型变量的值为 0xaa66。编译器是一个纯粹的 ANSl 编译器，写代码去完成这一任务

```
int *ptr;
 ptr = (int *)0x67a9;
 *ptr = 0xaa66;
```

188.【简答题】【嵌入式系统】中断是嵌入式系统中重要的组成部分 这导致了很多编译开发商提供一种扩展一让标准 C 支持中断。具代表事实是，产生了一个新的关键字 _interrupt。下面的代码就使用了_interrupt 关键字去定义了一个中断服务子程序（S 民），请评论一下这段代码的。

```
_interrupt double compute area (double raclius)
{
double area = PI * radius * radius;
Printf（" Area = %f",area);
return area;
```

```
}
```

a、ISR 不能返回一个值。

b、ISR 不能传递参数。

c、在许多处理器编译器中，浮点一般都是不可冲入的。有些处理器编译器需要让额外的寄存器入栈，有些处理器编译器就不允许在 ISR 中做浮点运算。此外 ISR 应该是短而有效率的。在 ISR 中做浮点运算是不明智的。

d、printf 经常是有冲入性和性能上的问题，所以一般不使用 printf 函数。

189.【简答题】【操作系统】什么是进程?什么是线程?两者有何区别?

进程是程序的一次执行，而进程又可以分为几个线程。一个进程中可以有多个线程，多个线程共享进程的堆和方法区 (JDK1.8 之后的元空间)资源，但是每个线程有自己的程序计数器、虚拟机栈 和 本地方法栈。

190.【单选题】【操作系统】已经获得除 CPU 以外的所有所需资源的进程处于（）状态

A 就绪状态

B 阻塞状态

C 运行状态

D 活动状态

答案：A

解析：进程的五状态模型：

运行态：该进程正在执行。

就绪态：进程已经做好了准备，只要有机会就开始执行。

阻塞态（等待态）：进程在某些事情发生前不能执行，等待阻塞进程的事件完成。

新建态：刚刚创建的进程，操作系统还没有把它加入到可执行进程组中，通常是进程控制块已经创建但是还没有加载到内存中的进程。

退出态：操作系统从可执行进程组中释放出的进程，或由于自身或某种原因停止运行。

191.【单选题】【操作系统】操作系统采用缓冲技术,通过减少对 CPU 的（A）次数,提高资源的利用率。

A 中断　　B 访问　　C 控制　　D 依赖

192.【简答题】【操作系统】简单叙述进程/线程之间的同步/互斥机制、及其实现方法

1 临界区:通过对多线程的串行化来访问公共资源或一段代码，速度快，适合控制数据访问。

2 互斥量:为协调共同对一个共享资源的单独访问而设计的。

3 信号量:为控制一个具有有限数量用户资源而设计。

4 事 件:用来通知线程有一些事件已发生，从而启动后继任务的开始。

193.【单选题】【云服务设计】FTP 服务和 SMTP 服务的端口默认分别是（C）

A 20 与 25

B 21 与 25

C 20，21 与 25

D 20 与 21

194.【多选题】【操作系统】 以下关于内存的说法正确的是

A RAM 是随机存储器，在断电时将丢失其存储内容，ROM 是只读存储器，断电时不会丢失存储内容

B 内存的数据带宽与内存的数据传输频率、内存数据总线位数以及内存大小有关

C 用户进程通常情况只能访问用户空间的虚拟地址,不能访问内核空间虚拟地址

D Linux 中使用 buddy system 算法可以管理页外内存碎片,使用 slub 算法可以管理页内内存碎片

答案：ACD

解析：B:内存的数据带宽的计算公式是：数据带宽=内存的数据传输频率×内存数据总线位数/8

195.【多选题】【计算机网络】下面协议中属于应用层协议的是（）

A ICMP、ARP

B FTP、 TELNET

C HTTP、SNMP

D SMTP、POP3

答案：BD

解析：

1、物理层：以太网 、 调制解调器 、 电力线通信(PLC) 、SONET/SDH 、 G.709 、 光导纤维 、 同轴电缆、 双绞线等。

2、数据链路层：Wi-Fi(IEEE 802.11)、WiMAX(IEEE 802.16) 、ATM 、 DTM 、 令牌环 、以太网、FDDI、帧中继、 GPRS 、 EVDO、HSPA 、HDLC 、 PPP 、 L2TP 、PPTP 、ISDN • STP、CSMA/CD 等。

3、网络层协议：IP IPv4 、IPv6、 ICMP、ICMPv6 • IGMP、IS-IS 、IPsec 、ARP 、RARP 、RIP 等。

4、传输层协议：TCP、 UDP、TLS 、 DCCP、 SCTP 、 RSVP 、OSPF 等。

5、应用层协议：DHCP 、DNS、 FTP 、Gopher 、 HTTP、 IMAP4 、 IRC、 NNTP 、 XMPP、POP3 、SIP、 SMTP、SNMP 、SSH、TELNET 、 RPC 、RTCP 、RTP 、RTSP、SDP 、 SOAP、GTP、STUN 、NTP、SSDP 、 BGP 等。

196.【简答题】【嵌入式系统】简述嵌入式系统的定义、应用和特点?

嵌入式系统的定义及特点

定义：嵌入式系统是以应用为中心、以计算机技术为基础，软、硬件可裁剪，适应于应用系统对功能、可靠性、成本、体积、功耗等方面有特殊要求的专用计算机系统。

特点：（1）嵌入式系统是面向特定应用的。嵌入式系统中的 CPU 是专门为特定应用设计的，具有低功耗、体积小、集成度高等特点，能够把通用 CPU 中许多由板卡完成的任务集成在芯片内部，从而有利于整个系统设计趋于小型化。

（2）嵌入式系统涉及先进的计算机技术、半导体技术、电子技术、通信和软件等各个行业。是一个技术密集、资金密集、高度分散、不断创新的知识集成系统。

（3）嵌入式系统的硬件和软件都必须具备高度可定制性。

（4）嵌入式系统的生命周期相当长。嵌入式系统和具体应用有机地结合在一起，其升级换代也是和具体产品同步进行的。

（5）嵌入式系统本身并不具备在其上进行进一步开发的能力。在设计完成以后，用户如果需要修改其中的程序功能，必须借助于一套专门的开发工具和环境。

（6）为了提高执行速度和系统可靠性，嵌入式系统中的软件一般都固化在存储器芯片或单片机中，而不是存贮于磁盘等载体中。想了解更多嵌入式只是可以去看看朱有鹏的免费视频，有关嵌入式的所有内容都有讲到。

197.【单选题】【数据结构与算法】list 和 vector 的区别有哪些（）

A vector 拥有一段连续的内存空间，因此支持随机存取，如果需要高效的随即存取，而不在乎插入和删除的效率，使用 vector.

B list 拥有一段不连续的内存空间，因此支持随机存取，如果需要大量的插入和删除，而不关心随即存取，则应使用 list

C 已知需要存储的元素时，使用 list 较好

D 如果需要任意位置插入元素，使用 vector 较好

答案：AB

解析：C:已知需要存储的元素时，vector 要好

D:如果需要任意位置插入元素，list 要好

198.【单选题】【数据结构与算法】若某线性表中最常用的操作是在最后一个元素之后插入一个元素和删除第一个元素,则最节省运算时间的存储方式是

A 单链表

B 仅有头指针的单循环链表

C 双链表

D 仅有尾指针的单循环链表

答案：D

单链表只能单向遍历，即只能由链表头向链表尾遍历。

单循环链表也只能单向遍历：链表头->链表尾->链表头；

对于 A，B，C 要想在尾端插入结点，需要遍历整个链表。

对于 D，要插入结点，只要改变一下指针即可，要删除头结点，只要删除指针.next 的元素即可。

如果只要知道尾指针 p，则通过计算一次 p->next 就能获得头指针；插入和删除算法复杂度 O(1)+O(1)

而如果只知道头指针，则需要遍历整个链表来获得尾指针的位置;插入和删除算法复杂度 O(1)+O(N)

所以 D 仅有尾指针的单循环链表存储方式最节省运算时间

199.【多选题】【计算机组成原理】下列说法正确的有

A 计算机体系结构是一门研究计算机系统软件结构的学科。

B 现代计算机处理器结构按照存储方式划分,可分为复杂指令集计算机和精简指令集计算机

C RISC 技术对比 CISC 最大的区别就是对 CPI 的精简

D 单指令流单数据流计算机的每个机器周期最多执行一条指令

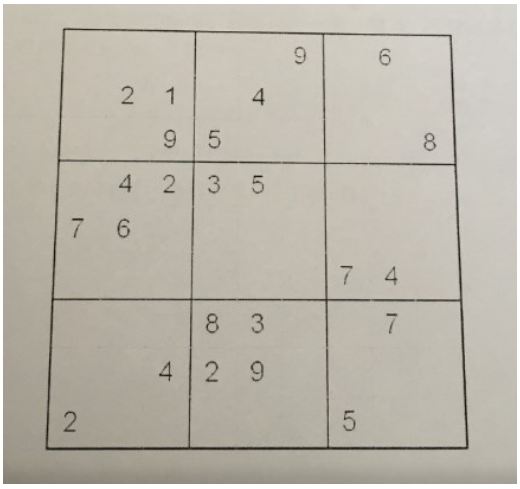
答案：CD

解析：

A、计算机体系结构主要研究软件、硬件功能分配和对软件、硬件界面的确定

B、现代计算机处理器结构按照指令系统方式划分,可分为复杂指令集计算机和精简指令集计算机

200.【简答题】【数据结构与算法】数独游戏是在 9x9 的方格内进行，用 1 至 9 之间的数字填满空格，一个格子填入一个数字，使其满足每一行、每列、每一个粗线宫内的数字均含 1-9，不重复。例如:



请设计算法，用程序来完成数独游戏，写明思路即可，不要求写代码。

思路，设计一个结构体，保存一个长度为 9 的整数数组（保存可能出现的数），一个标志位（同时代表可能的数的个数，当其为 1 时，代表以确定，0 表示需要推算），在使用递归依次实现每一个格子所能出现的可能，直到填满所有格子，或者发生冲突退出，并回馈错误。

201.【单选题】【操作系统】以下说法中那个不是进程和程序的区别（）：

A、程序是一组有序的静态指令进程是一次程序的执行过程
B、程序只能在前台运行，而进程可以在前台或后台运行
C、程序可以长期保存，进程是暂时的
D、程序没有状态，而进程是有状态的

答案：B。

解析：见计算机操作系统

202.【单选题】【操作系统】与通用操作系统相比嵌入式操作系统还必须具有的特点是（A）：

A、强稳定性，弱交互性
B、较强实时性

C、可伸缩性

D、功耗管理与节能

203.【填空题】【操作系统】以下为 Linux/windows 下的 32 位 C 程序，请计算 sizeof 的值

```
char str＝"Hello";
char *p= str;
int n＝ 10;
 sizeof(str）=___
 sizeof（p）=____
 sizeof（n）＝____

void Func (char str[100])
{
sizcof(str)=____;
}
void*p =malloc( 100)
sizcof(p)=____
```

6 4 4 4 4

在 32 位操作系统中，地址长度与位数相同为 4 个字节

204.【单选题】【权限操作】文件 exer1 的访问权限为 rw-r--r--，现要增加所有用户的执行权限和同组用户的写权限，下列命令正确的是（ ）

A、 chmod a+x g+w exerl B、 chmod 765 exerl

C、 chmod o+x exerl D、 chmod g+w exer1

答案：A。

解析：chmod ［who］ ［+|-|=］ ［mode］ 文件名¼

u 表示“用户（user）”，即文件或目录的所有者。

g 表示“同组（group）用户”，即与文件属主有相同组 ID 的所有用户。

o 表示“其他（others）用户”。

a 表示“所有（all）用户”。它是系统默认值。

操作符号可以是：

+ 添加某个权限。

- 取消某个权限。

= 赋予给定权限并取消其他所有权限（如果有的话）。

数字设定法的一般形式为：

chmod ［mode］ 文件名¼

我们必须首先了解用数字表示的属性的含义：0 表示没有权限，1 表示可执行权限，2 表示可写权限，4 表示可读权限，然后将其相加。所以数字属性的格式应为 3 个从 0 到 7 的八进制数，其顺序是（u）（g）（o）。

例如，如果想让某个文件的属主有“读/写”二种权限，需要把 4（可读）+2（可写）＝6（读/写）。

205.【单选题】【文件操作】在使用 mkdir 命令创建新的目录时,在其父目录不存在时先创建父目录的选项是（D）:

A、-m　B、-d　C、-f　D、　-P

206.【单选题】【计算机网络】下列提法中,不属于 ifconfig 命令作用范围的是:

A、配置本地回环地址

B、配置网卡的 IP 地址

C、激活网络适配器

D、加载网卡到内核中

加载网卡到内核中不属于 ifconfig 命令作用范围。因为：在 Linux 系统中，网卡的驱动程序是作为模块加载到内核中的，正因为如此，当没有网卡的驱动程序时，可以到网上去下载驱动程序的源文件，甚至可以自己动手编写网卡的驱动程序，然后以模块的形式将其编译到内核中去。

207.【单选题】【计算机网络】TCP/IP 协议模型中不包括下面哪项（B）:

A、物理层

B、网络接口层

C、 网络层

D、传输层

208.【单选题】【计算机网络】在进行协议分析时，为了捕获到流经网卡的全部协议数据，要使网卡工作在()模式下?:

A、广播模式

B、单播模式

C、混杂模式

D、多播模式

答案：C

解析：是指一台机器的网卡能够接收所有经过它的数据流，而不论其目的地址是否是它。

209.【单选题】【数据结构】若进栈序列为 1，2，3，4，进栈过程中可以出栈，则下列不可能的一个出栈序列是:

A、1，4，3，2　　B、2，3，4，1

C、3，1，4，2　　D、3，4，　2，1

答案：C

解析：栈为先进后出

210.【单选题】【计算机网络】IP 协议的特征是（C）:

A、可靠，无连接

B、不可靠,无连接

C、可靠,面向连接

D、不可靠，面向连接

211.【单选题】【符号连接文件】用 1s -al 命令列出下面的文件列表，哪一个文件是符号连接文件（A）。

A、 Irwxr--r-- 1 hel users 2024 Sep 12 08:12 cheng

B、drwxt-xr-X 3 yhf other 512 Sep 12 08:12 home

C、-rwxrwxrwx 2 hel-s users 56 Sep 09 11:05 goodbey

D、 -rw-rw-rw- 2 hel-s users 56 Sep 09 11:05 hello

212.【单选题】【计算机网络】ARP 协议工作过程中，当一台主机 A 向另一台主机 B 发送 ARP 查询请求时，以太网帧封装的目的 MAC 地址是:

A、源主机 A 的 MAC 地址　　B、目标主机 B 的 MAC 地址

C、任意地址：000000000000　D、广播地址：FFFFFFFFFFFF

答案：B

解析：ARP 为地址解析协议，它可以将 IP 地址转换为 MAC 物理地址

213.【简答题】【中断机制】如何理解中断的上半部和下半部

设备的中断会打断原有程序的正常运行，希望中断服务程序尽可能的短小。但事与愿违。因此将中断分为两种，上半部分（紧急的硬件中断）下半部分（延缓的耗时操作）

214.【简答题】【操作系统】操作系统有哪些管理功能？

进程管理，存储管理，设备管理，文件管理，作业管理

215.【简答题】【Android 操作系统】Android 中的上层应用程序通过哪些层访问底层设备节点？

应用程序框架层，系统运行层，linux 内核层

216.【简答题】【开发工具使用】Linux 编程中（Windows 编程也经常使用）调用系统 api 的时候会出现些错误，比方说使用 open(),write(),create()之类的函数有些时候会返回-1，也就是调用失败，这个时候往往需要知道失败的原因，如何快速知道失败的原因？

在调试器报错的地方，按住 ctrl 然后单击，实现跳转

217.【简答题】【操作系统，C 语言】__function__ __FILE__ __LINE__在 LINUX 下的 C/C＋编程中，这 3 个变量分别代表什么意思？自己有没有用过，什么情况下使用？

C/C++提供了三个宏__FUNCTION__，__FILE__和__LINE__来定位程序运行时的错误。程序预编译时预编译器将用所在的函数名，文件名和行号替换。当运行时错误产生后这三个宏分别能返回错误所在的函数，所在的文件名和所在的行号。

用来获取当前者个语句所在的位置，便于错误发现

218.【简答题】【实时操作系统】在某实时多任务操作系统中，有操作系统提供的 Sleep 函数。如下两种方式的延时，应选用哪种，为什么？

a）Sleep（1000）；

b）for （i＝0； i＜Performance＊1000；i＋＋）；

uCOSII、RTX 之类的操作系统，主要是通过定时器切换实现了多线程功能

裸机编程时，常常需要调用延时函数来进行等待。此时的 CPU 多处于闲置状态（例如执行 for 循环延时）。加入了实时操作系统后，RTOS 会利用定时器进行任务切换。在调用系统的延时函数时并非让 CPU 循环，而是判断是否有其他任务需要执行。从而提高 CPU 执行效率。但 RTOS 需要占用定时器。且会造成 RAM 消耗严重、实时性降低等问题

219.【单选题】【shell 编程】在 shell 编程中，下面哪个表示上一步运行程序的返回值（）

A、$#　B、$?　C、$&　D、$!

答案：B

解析：$# 表示 SHELL 脚本程序的命令行参数个数;

$? 表示 SHELL 脚本程序的上一命令返回值;

$! 执行上一个背景指令的 PID(后台运行的最后一个进程的进程 ID 号)

220.【单选题】【shell 编程】下面哪个 shell 语句不能打印出用户主目录的路径？（）

A、echo “$HOME”　B、echo ~　C、echo `$HOME`　D、echo $HOME

答案：C

解析：shell 规定单引号内的变量不进行转换，要输出变量的值，需要使用双引号

221.【单选题】【计算机网络】IP 地址 131.153.12.71 是一个（）类 IP 地址

A、A　B、B　C、C　D、D

答案：B

解析：A 类网络的 IP 地址范围为 1.0.0.1－127.255.255.254；

B 类网络的 IP 地址范围为：128.1.0.1－191.255.255.254；

C 类网络的 IP 地址范围为：192.0.1.1－223.255.255.254。

222.【单选题】【计算机网络】某网络的 IP 地址空间为 192.168.5.0/24，采用定长子网划分，子网掩码为 255.255.255.248，则该网络的最大子网个数、每个子网内最大可分配地址个数各为（）

A、8，32　B、32，8　C、32，6　D、8，30

答案：C

解析：一个 IP 的组成部分为： 网络号+子网号+主机号，

对于这样的 IP 192.168.5.0/24 由/24 可以知道其子网掩码默认为 255.255.255.0 ，255.255.255.248,二进制位：(11111111. 11111111. 11111111)(24 表示网络号).11111(子网掩码) 000 ，子网掩码中全 1 表示的网络号和子网号部分，0 部分表示主机号部分(11111)2 转换为十进制为 2^5 =32 所以可以划分为 32 个子网，而后面有 3 个 0，所以 000 - 111 表示的范围 2^3 -2（全 0 和全 1 不可用）=6

223.【单选题】【数据结构与算法】下列哪两个数据结构，同时具有较高的查找和删除性能？（）

A、有序数组　B、有序链表　C、AVL 树　D、Hash 表

答案：CD

解析：平衡二叉树的查找，插入和删除性能都是 O(logN) ，其中查找和删除性能较好；哈希表的查找、插入和删除性能都是 O(1) ，都是最好的。所以最后的结果选择： CD

224.【单选题】【数据结构与算法】下列排序算法中，哪些时间复杂度不会超过 nlog(n)？（AB）

A、快速排序　B、堆排序　C、归并排序　D、冒泡排序

225.【单选题】【数据结构与算法】判断有向图是否存在回路，利用（）方法最佳

A、拓扑排序　B、求最短路径　C、求关键路径　D、广度优先遍历

答案：A

解析：拓扑排序，每次选的点都是入度为 0 的点，如果没有入度为 0 的点，则不能构成拓扑排序，那么就存在回路

226.【单选题】【数据结构与算法】依次读入数据元素序列{a, b, c, d, e, f, g}进栈，元素进栈或出栈顺序是未知的，下列序列中，不可能成为栈空时弹出的元素构成序列的有（）

A、{d, e, c, f, b, g, a}

B、{c, d, b, e, f, a, g}

C、{e, f, d, g, c, b, a}

D、{f, e, g, d, a, c, b}

答案：D

解析：栈是先进后出，一旦选项中跳过了某一个直接进行他的前一个必错

227.【单选题】【数据结构与算法】下列有关图的遍历说法中，不正确的是（）

A、有向图和无向图都可以进行遍历操作

B、基本遍历算法两种：深度遍历和广度遍历

C、图的遍历必须用递归实现

D、图的遍历算法可以执行在有回路的图中

答案：C

解析：图的遍历分为递归和非递归实现，即为深度遍历和广度遍历

228.【简答题】【Linux 内核】insmod 命令会执行模块里的哪一个函数 rmmod 命令会执行模块里的哪一个函数，在执行 rmmod 命令时，是否存在卸载失败的情况？存在的话是由哪些问题引起的，该如何解决。

insmod 会调用 init_moudle 函数，rmmod 会调用 clean_moudle

存在：gcc 版本不对

229.【简答题】【Linux 内核】Linux 内核里面，内存申请有哪几个函数，各自的特点？

kmalloc 申请的内存位于物理内存映射区域，而且在物理上也是连续的，它们与真实的物理地址只有一个固定的偏移，因为存在较简单的转换关系，所以对申请的内存大小有限制，不能超过 128KB、

kzalloc 函数与 kmalloc() 非常相似，参数及返回值是一样的，可以说是前者是后者的一个变种，因为 kzalloc() 实际上只是额外附加了 __GFP_ZERO 标志。所以它除了申请内核内存外，还会对申请到的内存内容清零。

vmalloc 函数则会在虚拟内存空间给出一块连续的内存区，但这片连续的虚拟内存在物理内存中并不一定连续。由于 vmalloc() 没有保证申请到的是连续的物理内存，因此对申请的内存大小没有限制，如果需要申请较大的

内存空间就需要用此函数了。

230.【单选题】【Linux】which one of the following is used for the target filename in the makefile? (A)

A、$@　B、$*　C、$?　D、none of the mentioned

231.【单选题】【Linux】which option of gcc links with a library file?( )

A、-l　B、-link　C、none of the mentioned

答案：A:

解析：gcc 链接库文件的方式：1：直接在编译选项后加上库文件的绝对地址：2：通过 -l 选项

232.【多选题】【Linux】以下哪些事件会导致进程的创建

A、系统初始化

B、fork 系统调用

C、pthread_ create 函数调用

D、一个批处理作业的初始化

答案：ABD

解析：创建进程的多种方式但凡是硬件，都需要有操作系统去管理，只要有操作系统，就有进程的概念，就需要有创建进程的方式，一些操作系统只为一个应用程序设计，比如扫地机器人，一旦启动，所有的进程都已经存在。

而对于通用系统（跑很多应用程序），需要有系统运行过程中创建或撤销进程的能力，主要分为 4 中形式创建新的进程

1.系统初始化（查看进程 linux 中用 ps 命令， windows 中用任务管理器，前台进程负责与用户交互，后台运行的进程与用户无关，运行在后台并且只在需要时才唤醒的进程，称为守护进程，如电子邮件、web 页面、新闻、打印）

2.一个进程在运行过程中开启了子进程（如 nginx 开启多进程，os.fork 等）

3.用户的交互式请求，而创建一个新进程（如用户用鼠标双击任意一款软件，qq，微信等）

4.一个批处理作业的初始化（只在大型机的批处理系统中应用）

无论哪一种，新进程的创建都是由一个已经存在的进程执行了一个用于创建进程的系统调用而创建的。

233.【多选题】【Linux】Linux 执行 ls，会引起哪些系统调用（）

A、 nmap　　B、 read　　C、 execve　　D、 fork

答案：BC

解析：open 系统调用打开当前目录文件，返回获得的文件描述符。可以看到该文件使用 O_RDONLY 标志打开，只要该文件是用 O_RDONLY 或 O_RDWR 标志打开的，就可以用 read()系统调用从该文件中读取字节，所以 ls 要用到 read 系统调用。除此之外，任何 shell 命令都会创建进程，都会用到 exec 系统调用

234.【简答题】【Linux】32 位的 ARM Linux 上，应用程序可以使用的最大内存空间为多大?

4GB

235.【简答题】【Linux】mknod 命令的作用是什么?

mknod 命令用于创建 Linux 中的字符设备文件和块设备文件。

236.【简答题】【Linux】对于 gcc 编译器，命令 gcc 的参数-I -L 分别是做什么用的?

-l 参数就是用来指定程序要链接的库, -L 参数跟着的是库文件所在的目录名。

237.【简答题】【Linux】常见的 Makefile 中，关键字 CC、CPP、CFLAGS、CPPFLAGS、LDFLAGS 一般是做什么用的?

CC：C 语言编译器的名称；

CPP: C 语言预编译器的名称 ；

CFLAGS：C 语言编译器的编译选项

CPPFLAGS：C 语言预编译器的编译选项

LDFLAGS：gcc 等编译器会用到的一些优化参数，也可以在里面指定库文件的位置

238.【简答题】【Linux】请简述 Linux 操作系统上物理内存与虚拟内存有什么区别?

物理内存是真实存在的，它的表现形式是插在电脑主板上的内存条。看机器配置的时候，看的就是这个物理内存。而虚拟内存是虚拟存在的。它其实是电脑匀出一部分硬盘空间来充当内存使用。当内存耗尽时，电脑就会自动调用硬盘来充当内存，以缓解内存的紧张。

239.【单选题】【Linux】为了查看 linux 启动信息，可以用（B）

A、cat /etc/lilo.conf　　B、dmesg

C、cat /proc/cpuinfo　　D、lilo

240.【简答题】【Linux】内核态，用户态的区别

内核态，操作系统在内核态运行——运行操作系统程序

用户态，应用程序只能在用户态运行——运行用户程序

当一个进程在执行用户自己的代码时处于用户运行态（用户态），此时特权级最低，为 3 级，是普通的用户进程运行的特权级，大部分用户直接面对的程序都是运行在用户态。Ring3 状态不能访问 Ring0 的地址空间，包括代码和数据；当一个进程因为系统调用陷入内核代码中执行时处于内核运行态（内核态），此时特权级最高，为 0 级。执行的内核代码会使用当前进程的内核栈，每个进程都有自己的内核栈。

241.【简答题】【Linux】一段代码在什么情况下需要互斥？Linux 内核中如何实现互斥的？

实现控制某些系统资源在任意时刻只能允许一个进程访问的机制。互斥是同步机制中的一种特殊情况

(1)创建锁

// 创建互斥锁 mutex

pthread_mutex_t mutex;

(2)初始化锁

在 Linux 下, 线程的互斥量数据类型是 pthread_mutex_t 在使用前, 要对它进行初始化:

初始化的两种方法：(推荐使用第二种)

1.静态分配

pthread_mutex mutex = PTHREAD_MUTEX_INITIALIZER;

2.动态分配

int pthread_mutex_init(pthread_mutex_t *restrict mutex, const pthread_mutexattr_t *restrict attr);

mutex: 要初始化的互斥量(restrict 的作用是告诉调用者，不要改变指针的指向)

attr：锁的属性，一般写 NULL

加 restrict 的作用：只用于修饰函数参数里的指针，这个指针会频繁使用，所以把这个地址放到寄存器里，用着好找。

①设置线程的属性

int pthread_attr_init(pthread_attr_t *attr);//初始化线程属性

int pthread_attr_destroy(pthread_attr_t *attr);//销毁线程属性

Thread attributes(线程属性):

线程的分离属性: Detach state=PTHREAD_CREATE_DETACHED

线程的竞争范围: Scope = PTHREAD_SCOPE_SYSTEM

是否继承调度策略: Inherit scheduler = PTHREAD_EXPLICIT_SCHED

调度策略: Scheduling policy = SCHED_OTHER

调度优先级: Scheduling priority = 0

线程栈之间的保留区域: Guard size = 4096 bytes

自己指定栈地址: Stack address = 0x40197000

栈大小: Stack size = 0x3000000 bytes

//设置线程的分离属性

int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate);

//detachstate：有以下两种选择

PTHREAD_CREATE_DETACHED：设置成分离态

PTHREAD_CREATE_JOINABLE：设置成可结合态

//获取线程的分离属性

int pthread_attr_getdetachstate(pthread_attr_t *attr, int *detachstate);

//int *detachstate：输出型参数，将分离属性存放在该变量里

(3)上锁 && 解锁

242.【填空题】Linux】Linux 内核函数 Kmalloc 与 Vmalloc 之间的区别是_____Kmalloc 用于申请较小的、连续的物理内存,vmalloc()_用于申请较大的内存空间，虚拟内存是连续的_____

1. 以字节为单位进行分配，在<linux/vmalloC、h>中

2. void *vmalloc(unsigned long size) 分配的内存虚拟地址上连续，物理地址不连续

3. 一般情况下，只有硬件设备才需要物理地址连续的内存，因为硬件设备往往存在于 MMU 之外，根本不了解虚拟地址；但为了性能上的考虑，内核中一般使用 kmalloc()，而只有在需要获得大块内存时才使用 vmalloc()，例如当模块被动态加载到内核当中时，就把模块装载到由 vmalloc()分配 的内存上。

4.void vfree(void *addr)，这个函数可以睡眠，因此不能从中断上下文调用。

三.vmalloc()和 kmalloc()区别

1.kmalloc 保证分配的内存在物理上是连续的,那么它对应的虚拟内存肯定也是连续的,vmalloc 保证的是在虚拟地址空间上的连续,但物理内存不一定连续.由于 vmalloc()分配内存时,对应的物理内存是每个页框通过 alloc_page()来分配.

2.kmalloc 一般分配较小的内存,vmalloc 分配较大的内存.

3.vmalloc 比 kmalloc 要慢,且分配的虚拟地址空间位置不同.这两个函数所分配的内存虽然都处于内核空间(3GB～4GB),但对应的具体位置不同，kmalloc()分配的内存处于 3GB～high_memory 之间，而 vmalloc()分配的内存

在 VMALLOC_START～VMALLOC_END 之间，也就是非连续内存区。kmalloc 分配的内存,它的物理地址与虚拟地址只有一个 PAGE_OFFSET 偏移，不需要为地址段修改页表,而 vmalloc 分配内存时需要修改主内核页表.

一般情况下在驱动程序中都是调用 kmalloc()来给数据结构分配内存，而 vmalloc()用在为活动的交换区分配数据结构，为某些 I/O 驱动程序分配缓冲区，或为模块分配空间，例如在 include/asm-i386/module.h 中定义了如下语句：

#define module_map(x)　　vmalloc(x)

其含义就是把模块映射到非连续的内存区。

243.【简答题】【Linux】Linux 中新处理模块分成上半部分和下半部分的原因，为何要分？如何实现？

上半部分执行与硬件相关的处理要求快, 而有些驱动在中断处理程序中又需要完成大量工作,这构成矛盾,所以 Linux 有所谓的 bottom half 机制，中断处理程序中所有不要求立即完成的,在开中断的环境下,由底半程序随后完成. Linux 的底半处理实际上是建立在内核的软中断机制上的.

Linux 的底半 机制主要有 Tasklet 和 work queue 以及 softirq (2.4 内核则有 BH , Task queue , softirq , tasklet 没有 work queue)，其实底半可以理解成一种工作的延迟。所以实际使用时跟 timer 机制基本上一个意思

244.【简答题】【Linux】LINUX 进程的查看命令和时间查看命令。

ps,-ef

245.【简答题】【Linux】简述 socket 编程中，udp 协议服务器和客户端调用的 API 的过程。

客户端：

1. 创建通信的 UDP 套接字

2. 定义一个 IPv4 的结构，存放服务器的地址信息

服务端：

创建通信的 UDP 套接字

定义 IPv4 结构，存放本机信息

给 UDP 套接字 bind 一个固定的地址

246.【单选题】【C 语言】以下程序运行结果是（B）

```
void    foo(int *a,    int *b)
{
*a = *a + *b;
```

```
*b = *a - *b;
*a = *a - *b;

}
void main( )
{
int a= 1, b = 2, c = 3;
foo(&a, &b);
foo(&b, &c);
foo(&c, &a);
printf(“%d, %d, %d”, a, b, c);
}
```

A、1，2，3

B、1，3，2

C、2，1，3

D、3，2，1

247.【单选题】【C 语言】下列一段 C++代码的输出是？（B）

```
#include “stdio.h”
class Base
{
public:
int Bar(char x)
{
return (int)(x);
}

virtual int Bar(int x)
{
return (2*x);
}
};

class Derived：public Base
```

```
{
public:
int Bar(char x)
{
return (int)(-x);
}
Int Bar(int x)
{
return (x/2);
}
};

int main(void)
{
Derived Obj;
Base *pObj = &Obj;
printf(“%d”, pObj->Bar((char)(100)));
printf(“%d”, pIObj->Bar(100));
}
```

A、100，-100

B、100，50

C、200，-100

D、200，50

248.【单选题】【C 语言】下面 C++程序的输出是（B）

```
void f(char *x)
{
x++;
*x = ‘a’;
}

int main()
{
char str[sizeof(“hello”)];
```

```
strcpy(str, “hello”);
f(str);
cout<<str;
return 0;
}
```

A、hello　B、hallo　C、allo　D、以上都不是

249.【单选题】【C 语言】若 a 是 int 类型变量，则计算表达式 a=25/3%3 的值是（B）：

A、3　B、2　C、1　D、0

250.【单选题】【C 语言】C 语言规定，简单变量作为实参时，他和对应形参之间的数据传递方式是（A）：

A．单向值传递　B、地址传递　C、相互传递　D、由用户指定方式

251.【简答题】【C 语言】关键字 static 和 volatile 各有什么含义？

volatile 防止编译器优化代码

Static

（1）在修饰变量的时候，static 修饰的静态局部变量只执行初始化一次，而且延长了局部变量的生命周期，直到程序运行结束以后才释放。

（2）static 修饰全局变量的时候，这个全局变量只能在本文件中访问，不能在其它文件中访问，即便是 extern 外部声明也不可以。

（3）static 修饰一个函数，则这个函数的只能在本文件中调用，不能被其他文件调用。static 修饰的变量存放在全局数据区的静态变量区，包括全局静态变量和局部静态变量，都在全局数据区分配内存。初始化的时候自动初始化为 0。

（4）不想被释放的时候，可以使用 static 修饰。比如修饰函数中存放在栈空间的数组。如果不想让这个数组在函数调用结束释放可以使用 static 修饰。

（5）考虑到数据安全性（当程序想要使用全局变量的时候应该先考虑使用 static）。

252.【编程题】【C 语言】输入任意字符串，打印输出其逆序。

```
#include <stdio.h>
#include<stdliB、h>
```

```
#include<string.h>
int main()
{   int len,i;
    char a[100];
    gets(a);
    len=strlen(a);
    for(i=len-1;i>=0;i--)
        printf("%c",a[i]);

    return 0;

}
```

253.【编程题】【C 语言】下面程序完成以下功能 trans()函数将二维数组 a 进行转置，即行列互换，如下图所示：

1 2 3 4　　1 2 5 7
2 4 6 8　　2 4 8 1
5 8 9 7　　3 6 9 6
7 1 6 5　　4 9 7 4
转置前　　转置后

```
#include <stdio.h>
#define M 3
#define N  4
int Transpose(int arry1[][N],int arry2[][M])          //转置函数
{
    for(int i=0;i<M;i++)
        for(int j=0;j<N;j++)
            arry2[j][i]=arry1[i][j];
    return 0;
}

int main()
{
    int arry1[M][N],arry2[N][M],i,j;
    printf("请输入转置前二维数组\n");
```

```
    for(i=0;i<M;i++)
        for(int j=0;j<N;j++)
            scanf("%d",&arry1[i][j]);
    Transpose(arry1,arry2);                    //转置
    printf("转置后\n");
    for(i=0;i<N;i++)
    {
        for(int j=0;j<M;j++)
            printf("%d\t",arry2[i][j]);
        printf("\n");
    }
    return 0;
}
```

254.【简答题】【C 语言】嵌入式常对变量或寄存器进行位操作，给定一个 int 型 变量 a，写两段代码:1.设置 a 的 bit3，2.消除 a 的 bit3 以上操作需要其他位保持不变考虑可移植性。

```
Int a ;
Int temp = 1<<(3)
A =a&temp;
//
Int a;
Int temp = ~(1<<3);
A=a&temp;
```

255.【简答题】【C 语言】请分别说明 strlen 和 sizeof 函数的区别，并实现 strlen 函数

sizeof 是得到某个类型所占的字节数，strlen 得到的是一个字符串从开头地址到结束符的长度

实现：

```
Int strlen(char*data)
{
Int number =0 ;
While(data[number]!=’\0’)number++;
Return number;
```

```
}
```

256.【简答题】【C 语言】C 语言中的关键字 const,volatile,inine 分别是做什么用的?

关键字 const 用来定义只读变量。Volatile 说明程序在执行中可被隐含的改变。inline 关键字用来定义一个类的内联函数。

257.【简答题】【C 语言】Using C create a 1GB size file

使用 malloc 函数 申请 1 个 GB 的空间大小

由于 1GB=1073741824 字节。所以为：

char *one_gb;

one_gb=(char *) malloc(107374182);

258.【编程题】【C 语言】写一个宏 MIN，这个宏输入两个参数并返回较小的一个

```
#define Min(a,b) ( ((a)>=(b))?(b):(a))
```

259.【编程题】【C 语言】给定一个整型变量 data,写两段代码，第一个设置 data 的 bit5, 第二个清除 data 的 bit 5.注意保持其它位不变。

```
#define BIT5 (0x1 << 5)
static int a;
void set_bit5(void)
{
    a |= BIT5;
}
void clear_bit5(void)
{
    a &= ~BIT5;
}
```

260.【简答题】【C 语言】C++里父类和子类之间有什么关系? friend 类有什么特点?

父类和子类是继承关系。

三种继承方式

1.public 继承；

2.protect 继承；

3.private 继承；

261.【简答题】【C 语言】写一个函数用来找出字符串中第一个无重复的字符。例如，“total”中第 1 个无重复的字符为 o, "teeter"中第一个无重复字符为 r。

```
#include <stdio.h>
#include <string.h>

char findIt(const char *str);
int main()
{
    char str[100]/*="teeter"*/;

    scanf("%s",str);

    printf("%c\n",findIt(str));

    return 0;
}

char findIt(const char *str)
{
    int count[26]={0};
    int index[26]={0};  //注意 int 数组初始化赋值时，如果写成={-1}是不能给所有元素初始化为-1 的，只有第一个元素是-1，其余为默认值 0
    unsigned int i;
    int pos;
    for(i=0;i<strlen(str);i++)
    {
        count[str[i]-'a']++;    //记录该字母出现的次数
        // cout<<count[str[i]-'a']<<endl;
        if(index[str[i]-'a']==0)
        {
            index[str[i]-'a']=i;  //记住该字母第一次出现时的索引
```

```
        }
    }
    pos=strlen(str);
    for(i=0;i<26;i++)
    {
        if(count[i]==1)   //找到只出现一次的字母
        {
            if(index[i]!=-1&&index[i]<pos)   //在只出现一次的字母中找出索引值最小的即可
            {
                pos=index[i];
            }
        }
    }
    if(pos<strlen(str))
        return str[pos];
    return '\0';
}
```

262.【简答题】【C 语言】volatile 关键字的意义是什么?我们在使用过程中如果遇到 volatile 类型该如何规避风险?

volatile 关键字是一种类型修饰符，用它声明的类型变量表示可以被某些编译器未知的因素更改，比如：操作系统，硬件或者其他线程等。

少用 volatile ，因为 volatile 只会降低代码的性能。

263.【简答题】【C 语言】描述以下代码的区别

char a[] = “Hello world”

char *s = “Hello world”

a 为创建一个数组，而 s 为创建一个指针。

264.【简答题】【C 语言】说明 int(*p)[5]与 int *p[5]的区别

int *p[5],首先它是一个数组，它的大小是 5，它里面存放的数据类型是 int *,也就是整型指针。 所以它叫指针数组、 （从右向左解析这个表示）

int (*p)[5]，首先 p 是一个指针，指向大小为 5 的数组，因此这叫数组指针。

265.【简答题】【C 语言】给出一个整数，请写出一个函数，判断这个数如果不是小于 100 的非负整数，就返回-1，否则返回 0，并写代码验证缩写的函数是否正确；

```
#include <iostream>
using namespace std;
void text1 ()
{
	int a;
	cin>>a;
	if(a>=0&&a<100)
	{
		cout<<0;
	}
	else {
		cout<<-1;
	}
}
int main()
{
	text1();
	return 0;
}
```

266.【简答题】【C 语言】请找出下面代码中的所有错误

说明；以下代码是把一个字符串倒序，如“abcd”变为“dcba”

```
Main()
{
Char *src=”hello,world”;
Int len=strlen(src);
Char *dest=(char *)malloc(len);
Char *d=dest;
Char *s=src[len];
```

```
While(len--!=0)
{
d++=s--;
}
Printf(“%s\n”,dest);
Return 0;
}
```

[警告]不推荐将字符串常量转换为“char； 未添加 头文件 #include <string.h> 文件；左值作为赋值的左操作数。

267.【简答题】【C 语言】上题中如果字符串倒述之后，原字符串没有用了，请问有什么其他的方式实现字符串倒序，可以写多种，并请指出各个方法的优缺点。

一：从第二项起，将后面的元素与它前面的元素逐个交换位置。

二：双指针法，使用两个指针分别指向字符串头和尾

三、递归入栈： 我们知道，在栈区，变量是先进后出的，那么我们可以这样做来逆序字符串。

268.【单选题】【C 语言】32 位系统中，该程序的输出为

```
//参数传递 退化为指针
void Func(char str_arg[100])
{
    printf("%d\n",sizeof(str_arg));
}
int main()
{
    char str[] = "Hello";
    printf("%d\n",sizeof(str));
    printf("%d\n",strlen(str));
    char *p = str;
    printf("%d\n",sizeof(p));
    Func(str);
    return 0;
}
```

A 5 5 4 4

B 6 5 4 4

C 6 5 6 4

D 5 5 5 100

答案：B

解析：使用函数**strlen()求某个字符串的长度时是不包括结尾标志符'\0'的，但当你用 sizeof()**求某个字符串占用的内存空间时，结尾字符'\0'是被包括在里面的。

strlen 用来计算字符串的长度（在 C/C++中，字符串是以"\0"作为结束符的），它从内存的某个位置（可以是字符串开头，中间某个位置，甚至是某个不确定的内存区域）开始扫描直到碰到第一个字符串结束符\0 为止，然后返回计数器值。

sizeof 是 C 语言的关键字，它以字节的形式给出了其操作数的存储大小，操作数可以是一个表达式或括在括号内的类型名，操作数的存储大小由操作数的类型决定

269.【单选题】【C 语言】int i =1;const int j =2;以下说法不正确的是

A const int *p1 = &i;

B const int *p2 = &j;

C int *const p3 = &i;

D int *const p4 = &j;

答案：D

int *const p4 ,p4 为指针常量，p4 指向的内存位置不能改变，但是，p4 所指内存存放的值是可以改变的。j 表示常量，其数值不能被改变。

将 j 的地址赋给 p4 后，p4 可以执行其他操作（ 如*p4=4;），将 j 的值改变，因此，int *const p4 = &j;是错的。

270.【单选题】【C 语言】有以下程序,求输出结果

```
#include<stdio.h>
int fun(int i)
{
    int cnt = 0;
    while(i)
    {
        cnt++;
        i=i&(i-1);
    }
```

```
    return cnt;
}
int main()
{
    printf("%d\n\r",fun（2021));
    return 0;
}
```

答案：8

解析：&是按位与，对应位都为 1 时该位得 1，否则得 0。所以 i&(i-1) 的作用：将 i 的二进制表示中的最右边的 1 置为 0。

在本题中即数出 2021 转换成二进制有几个 1 就会走几次循环(不断除 2)。2021 对应的二进制是：10100111111，一共 8 个 1，故走 8 次。

扩展：(n > 0 && ((n & (n - 1)) == 0)是判断 n 是不是 2 的次幂

271.【单选题】【C 语言】若 int x = 5&6,那么 x 的值为（）

A、 3　　B、 4　　C、 5　　D、 6

答案：B

解析：5: 0101

6: 0110

X:0100

272.【单选题】【C 语言】以下错误的表达式为

```
struct {
    inr a;
    char b;
}Q,*p=&Q;
```

A Q.a

B (*p).b

C p->a

D *p.b

D

*p=&Q ，把 Q 的地址赋值给了指针 p，对 p 解引用其实就是 Q。

A 选项肯定是对的，结构体的正常访问方法。

B 选项 (*p).b 等价于 Q.b

C p->a p 为指针访问结构体用->没问题。

D *p.b 优先级问题，.的优先级高于 *，所以 *p.b == *（p.b），p 为指针，访问结构体成员要用->。

优先级	运算符	结合性
1	() [] .	从左到右
2	! +(正) -(负) ~ ++ --	从右向左
3	* / %	从左向右
4	+(加) -(减)	从左向右
5	<< >> >>>	从左向右
6	< <= > >= instanceof	从左向右
7	== !=	从左向右
8	&(按位与)	从左向右
9	^	从左向右
10	|	从左向右
11	&&	从左向右
12	||	从左向右
13	?:	从右向左
14	= += -= *= /= %= &= |= ^= ~= <<= >>= >>>=	从右向左

扩展：结构体中.和->两种访问区别

定义结构体指针，访问成员时就用->

定义结构体变量，访问成员时就用.

```
struct A {
int a;
char b;
};
struct A q; //访问成员就用：q.a;
struct A *p; //访问成员就用：p->a;
```

273.【多选题】【C 语言】下列叙述正确的是（）

A 指针可以为空，引用不能为空。

B 不存在指向空值的引用，但是存在指向空值的指针

C 引用必须被初始化，但是指针不必

D 指针初化后不能被改变，引用可以改变所指对象

答案：ABC。

解析：指针初始化后能够被改变，引用初始化后不能被改变。

274.【简答题】【C 语言】已知 strcpy 的函数原型：char *strcpy(char *strDest, const char *strSrc), 其中 strDest 是目的字符串，strSrc 是源字符串， 不用调用 C 的字符串函数，写出函数 strcpy。（C 试题）

```
#include <stdio.h>
#include <cstring>

char *mystrcpy(char *strDest, const char *strSrc) {
    if (strDest == NULL || strSrc == NULL) return NULL;
    if (strDest == strSrc) return strDest;
    char *tempptr = strDest;
    while((*strDest++ = *strSrc++) != '\0');
    return tempptr;
}

int main() {
    char str1[10] = "hello";
    char str2[10];
    mystrcpy(str2, str1);
    printf("%s\r\n", str2);
    return 0;
}
```

275.【简答题】【C 语言】用指针的方法，将字符串 ”ABCD1234efgh”前后对调显示。

```
#include<stdio.h>

char *reverse(char *str) {
    int strsize = 0;
```

```
    for (int i = 0; str[i]; i++) strsize = i;
    char *left = str, *right = &str[strsize];
    while(left < right) {
        *left ^= *right;
        *right ^= *left;
        *left ^= *right;
        left++, right--;
    }
    return str;
}

int main() {
    char str1[20] = "ABCD1234efgh";

    printf("%s\r\n", str1);
    reverse(str1);
    printf("%s\r\n", str1);
    return 0;
}
```

276.【简答题】【C 语言】用 C 语言实现字符串转数字函数：

```
int atoi（const char *str）。
#include<stdio.h>

int my_atoi(char *str)
{
    int i = 0;
    /** 定义标志位 flag，设该字符串为正 */
    int flag = 1;
    int value = 0;
    if('-' == str[0])
    {
        /** 字符串第一个字符为‘-’,则将 flag 置为-1 */
        flag = -1;
        i++;
```

```
        }
        else if('+' == str[0])
        {
                i++;
        }
        /**  碰到不是数字的字符则结束循环  */
        while(str[i] >= '0'    && str[i] <= '9')
        {
                value = value*10 + str[i] - '0';
                i++;
        }

        return flag*value;
}

int main(void)
{
       int a = 0;
        char str1[] = "+123"; //输出 123
       a = my_atoi(str1);
       printf("%d\r\n", a);

        char str2[] = "-123";  //输出-123
       a = my_atoi(str2);
       printf("%d\r\n", a);
        char str3[] = "123";   //输出 123
       a = my_atoi(str3);
       printf("%d\r\n", a);
        char str4[] = "123asd45";   //输出 123
       a = my_atoi(str4);
       printf("%d\r\n", a);

        return 0;
}
```

277.【简答题】【C 语言】请问运行 Test 函数会有什么样的结果

```
void GetMenory(char *p)
{
    p= (char *)malloe(100);
}
void Test(void)
{
char *str= NULL;
GetMemory(str)
strepy(str,"hello world);
printf(str);
}
```

输出“hello world”，melloc 是 C 语言中的标准库函数，strcpy 同样

278.【简答题】【C 语言】头文件中的#ifndef/define/endif 的作用。

条件编译

279.【简答题】【C 语言】请问 int *p 和 char *p 分别占几个字节？为什么？

一个指针在 32 位的计算机上,占 4 个字节; 一个指针在 64 位的计算机上,占 8 个字节。

280.【简答题】【C 语言】有一数组，int a[4] = {1,2,3,4}； *((int *)(&a + 1) - 1)的结果是：

4

&a + 1 到数组 a[3]之后的一个元素，转换成 int *类型后-1 到 a[3];

281.【简答题】【C 语言】设地址为 0x100 的整形变量的值为 2000，c 语言下如何实现

#define MYADDR 0x10

Int *p = (int *)MYADDR;

P = 2000;

282.【简答题】【C 语言】写一个宏函数，功能：求结构体成员相对于其首地址的偏移地址

#define STRUCT_OFFSET(id, element) ((unsigned long) &((struct id*) 0)->element)

283.【简答题】【C 语言】请简述 sizeof 与 strlen 的区别

strlen 计算字符串的长度，以'\0'为字符串结束标志

sizeof 是分配的数组实际所占的内存空间大小，不受里面存储内容

284.【简答题】【C 语言】嵌入式系统中经常要用到无限循环，你怎么样用 c 编写死循环呢？

while(1);

285.【简答题】【C 语言】完成字符串拷贝可以使用 sprintf、strcpy 及 memcpy 函数，请问这些函数有什么区别，你喜欢使用哪个，为什么？

sprintf 可以进行额外的格式化

strcpy 会复制直到出现 " 为止，可能溢出

strncpy 会复制一个以 " 结束的字符串，但是如果字符串长度超过指定数量则被截断，但结果可能不包含 " 表示结束

memcpy 只负责复制指定数量的 bytes，不处理 " 的情况

286.【简答题】【C 语言】static 有什么用途？（请至少说明两种）

1. 限制变量的作用域
2. 设置变量的存储域

287.【简答题】【C 语言】Volatile 一般用在什么地方，int *volatile reg、volatile int *reg volatile int *volatile

volatile 关键字告诉编译器，这个变量可能在任何时刻都会被改变。有时变量的改变对编译器来说有点“隐蔽”，也许就是这种隐蔽让编译给我们的程序造成了一些未知的 bug。

288.【简答题】【C 语言】用变量 a 给出下面的定义

a)一个整形数

b)一个指向整形数的指针

c)一个指向指针的指针，他指向的指针是指向一个整形数的

d)一个有 10 个整形数的数组

e)一个有 10 个指针的数组，该指针是指向一个整形数的

f)一个指向有 10 个整形数数组的指针

g)一个指向函数的指针，该函数有一个整形参数并返回一个整形数

h)一个有 10 个指针的数组，该指针指向一个函数，该函数有一个整形参数，并返回一个整形数

a) int a;

b). int *a;

c).int **a;

d). Int a[10];

e). Int *a[10];

f). int (*a)(10);

g). Int (*a)(int);

h). Int (*a[10])(int);

289.【简答题】【C 语言】关于 float x 与零值比较的 if 语句。

a)x!=0

b)X == 0

const float EPSINON = 0.00001;

if ((x >= -EPSINON) && (x <= EPSINON))

//EPSINON 是允许的误差,只要在这个范围内就说明 x 与 0 是相等的。

290.【简答题】【C 语言】给定一个整型变量 a，写两段代码，第一个设置 a 的 bit 3,第二个清除 a 的 bit 3. 在以上的两个操作中，要保持其他位不变。

```
#define BIT3 (0x1 << 3)
static int a;
void set_bit3(void)
```

```
{
    a |= BIT3;
}
void clear_bit3(void)
{
    a &= ~BIT3;
}
```

291.【简答题】【C 语言】下面的声明都是什么意思？

Const int a;

Int const a;

Const int *a;

Int *const a;

Int const *a const;

Const int a; int const a;这两个写法是相同的，表示 a 是一个 int 常量。

Const int *a; 表示 a 是一个指针，可以任意指向 int 常量或者 int 变量，他总是把它所指向的目标当作一个 int 常量。也可以写成 int const *a；含义相同。

Int *const a ;表示 a 是一个指针常量，初始化的时候必须固定一个指向 int 变量，之后就不能再指向别的地方。

Int const *a const;这个写法没有，倒是可以写成 int const * const a;表示 a 是一个指针常量，初始化的时候必须固定指向一个 int 常量或者 int 变量，之后就不能再指向别的地方了， 它总是把它所指向的目标当作一个 int 常量。也可以写成 const int * const a;

292.【简答题】【C 语言】请问一下程序有没有问题，如果有请写出为什么。

```
void main(void)
{
Char *p = “welcome to vguang”;
Char data[16];
Strcpy(data, p);
Printf(“data is %s\n”, data);
}
```

有问题。Data 的长度不足以存放 p 指针指向的对象。

293.【简答题】【C 语言】请写出以下程序的执行结果

```
Int a = 1;
Int b = a++;
Int c = a++;
If (a &b ++ || c++)
Printf(“%d-->%d-->%d\n”, a, b, c);
```

3->2->2

294.【简答题】【C 语言】请写出以下程序的执行结果

```
Void test (char c)
{
Uint8_t buf[10];
buf[0] = 0x95;
If(c == buf[0]) {
Printf(“1\n”);
} else {
Printf(“2\n”);
}
}
void main (void)
{
Char data = 0x95;
test(data);
}
```

2

Uint8_t 和 int8_t 的区别

295.【简答题】【C 语言】请问一下程序是否可以正常运行，若不能，请写出理由

```
Uint8_t *test(int *m)
{
Uint8_t buf[256];
```

```
Memcpy(buf ,m 10);
Return buf;
}
Void main(void)
{
Char buf[1024] = “welcome to vguang”;
Char *str = test(buf);
Printf(“str = %s\n”, str);
}
```

不行：test 函数内的 buf 作用域为 test 函数内部，函数返回，空间被回收。

296.【简答题】【C 语言】请问一下程序是否可以正常运行，若不能，请写出理由

```
Void test(void)
{
Char buf[128];
Char *p = buf;
For(i=0;i<256;i++){
*p++ = i;
}
}
Void main(void)
{
Test();
}
```

不能够运行。在 for 循环中并没有声明 i 的类型，且超出了 buf 数组的边界

297.【填空题】【C 语言】完成填空题

```
# include <stdio.h>
void main( )
char s1[]="Hello World!";
char s2[]="Hello World!";
if(s1==s2)
printf("Equal!");
```

```
else
printf(“Not equal!”);
}
```

输出结果_____

Not equal!

s1 和 s2 保存的是两个字符串的首地址，地址并不相同

298..【填空题】【C 语言】完成填空题

```
# include <stdio.h>
void main()
{
char s[="Hello World!";
char*p=s;
int n=10;
printf("%d %d %d %dn" ,sizeof(s),sizeof(p),strlen(p)，sizeof(n));
}
```

输出结果_____ _____

13 8 12 4n

sizeof(s)是所保存的字符串+‘\0’结束，sizeof（p）p 是一个字符串首地址，在六十四位操作系统中，地址为六十四位，8 个字节。Strlen，从字符串头地址开始计算，知道访问到‘\0’结束符停止计数。Int 占四个字节。

```
.【填空题】【C 语言】完成填空题
#include <stdio.h>
void swap (int *p1,int *p2)
{
int t;
int* p;
t= *p1;
*pl= *p2;
*p2=t;
p=pl;
p1=p2;
p2=p;
```

```
}
void swap2(int **m, int** n)
{
int *p;
P=*m;
*m=*n;
*n=p;
}
   void main( )
   {
int a=1;b=2;*p=&a,*q=&b;
        swap(p,q):
        printf(“%d,%d,%d,%d,"a,b,*p,*q);
        swap2(&p,&q);
        printf(“%d,%d,%d,%d\n,"a,b,*p,*q);
}
```

输出結果:_____ _____

2,1,2,1

2,1,1,2

swap 函数作用交换两个指针内的数据，swap2 函数作用将 p 指针指向的值复制给 p2 所指向的 内存

299.【填空题】【C 语言】完成填空题

```
# include <stdio.h>
   void merge (char *d, int size,char*sl,char*s2)
   {
while(*sl !=O && *s2!=0)
              {
if (*s1<*s2)
                          *d++=*s1++;
                   else
d++= *s2++;
}
   while(*s1!=0) *d++= *sl++;
while(*s2!=0) *d++= *s2++;
```

```
   *d=0;
}
void main( )
{
char s1[]="acmghn",s2[]="bcfhi",s3[20];
memset(s3,0,sizeof(s3));
merge(s3,sizeof(s3)-1,s1,s2);
puts(s3);
}
```

输出结果: _____

abccfhimghn

merge 函数，依次比较 s1 和 s2 字符串中较大的字符放入 s3 中，最后，吧剩下的放入 s3 的末尾

300.【单选题】【c 语言+Linux】

```
class A
{
int a;
short b;
int c;
char d;
};
class B
{
double a;
short b;
int c;
char d;
};
```

在 32 位机器上用 gcc 编译以上代码，求 sizeof(A)，sizeof(B)分别是多少（）

A、12 16

B、12 12

C、16 24

D、16 20

D

其实不同编译器答案是不一样的。

gcc 下编译答案是 D

linux 下 double 的对齐方式是 4 字节，而 windows 默认是 8 字节。。。

linux 可以通过如下方式控制对齐方式。

而 WINDOWS 是通过

#pragma pack(n)　来控制编译器按照 n 个字节对齐。

但是如果按照大家比较能够理解的说法：

根据以下条件进行计算：

1、 结构体的大小等于结构体内最大成员大小的整数倍

2、 结构体内的成员的首地址相对于结构体首地址的偏移量是其类型大小的整数倍，比如说 double 型成员相对于结构体的首地址的地址偏移量应该是 8 的倍数。

3、 为了满足规则 1 和 2 编译器会在结构体成员之后进行字节填充！

那么答案就是：C

301.【单选题】【C++】下列关于 C++容器描述错误的是？（）

A list 类型支持双向顺序访问，在 list 中任何位置插入删除都很快

B deque 类型支持快速顺序访问，在头尾插入／删除速度很快

C C++标准库 map 的底层实现为红黑树

D vector 类型在每次调用 pushback 时都在栈上开辟新内存

答案：B。

解析：deque 类型支持快速随机访问，在头尾插入/删除速度都很快。

302.【多选题】【C++】C++中，下列数据类型的转换，哪个可能会发生信息丢失？

A int -> double

B int -> long

C long -> float

D int -> char

答案 CD

精度丢失只会发生在从大范围到小范围的转换

32 位编译器：

char short int long float double 指针

1 2 4 4 4 8 4

64 位编译器：

char short int long float double 指针

1 2 4 8 4 8 8

303.【单选题】【C++】关于对象的 this 指针,以下叙述不正确的有

A 必须显示地在类中定义声明 this 数据成员才能使用 this 指针

B 一旦生成一个对象,该对象的 this 指针就指向该对象本身

C 一个类的所有对象的 this 指针的值都是相同的

D 不能通过对象的 this 指针访问对象的数据成员和成员函数

答案：A

解析：this 指针的特点：

（1）每个当前对象都含有一个指向该对象的 this 指针。this 指针只能在类的成员函数中使用，在全局函数、静态成员函数中都不能使用 this 。

（2）this 指针是在成员函数的开始前构造，并在成员函数的结束后清除 。

（3）this 指针会因编译器不同而有不同的存储位置，可能是寄存器或全局变量 。

（4）this 是类的指针 。

（5）因为 this 指针只有在成员函数中才有定义，所以获得一个对象后，不能通过对象使 用 this 指针，所以也就无法知道一个对象的 this 指针的位置。 不过，可以在成员函数中指定 this 指针的位置 。

（6）普通的类函数（不论是非静态成员函数，还是静态成员函数）都不会创建一个函数表来保存函数指针，只有虚函数才会被放到函数表中。

304.【简答题】【体系与内核基础】设备文件/dev/mem 是做什么用的?它属于什么么类型的设备？一般如何进行读写操作?

“/dev/mem”设备是内核所有物理地址空间的全映像；

物理内存（RAM）空间

物理存储（ROM）空间

cpu 总线地址

cpu 寄存器地址

外设寄存器地址，GPIO、定时器、ADC

“/dev/mem”设备通常与“mmap”结合使用，将该设备的物理内存映射到用户态，这样用户空间可以直接访问内存态。

使用方法：

“/dev/mem”设备通常与“mmap”结合使用，将该设备的物理内存映射到用户态，在用户态直接访问内核态

物理内存。 因为涉及访问内核空间，因此只有 root 用户才有访问“/dev/mem”设备的权限。

第一步，open 一个“/dev/mem”文件描述符，访问权限可以为只读（O_RDONLY ）、只写（O_WRONLY ）、读写（O_RDWR ）的阻塞或者非阻塞方式。

第二步，通过 mmap 把需访问的目标物理地址与“/dev/mem”文件描述符建立映射。

第三步，地址读写访问。

305.【填空题】【体系与内核基础】GPIO 支持_____三种数据传输方式

高电平、低电平、和高阻状态

306.【填空题】【体系与内核基础】总线通常分为三类，分别是_____。

控制总线、地址总线和数据总线

307.【简答题】【体系与内核基础】简述大端和小端字节序？X86 系列 CPU 是什么字节序？

小端字节序指低字节数据存放在内存低地址处，高字节数据存放在内存高地址处；大端字节序是高字节数据存放在低地址处，低字节数据存放在高地址处。 基于 X86 平台的 PC 机是小端字节序的，而有的嵌入式平台则是大端字节序的。

308.【简答题】【体系与内核基础】嵌入式设备，为加快启动速度，可以做哪些方面的优化?

linux 默认的安装内核相当庞大，为了保证系统的兼容性和灵活性，支持热插拔操作，内核启动时要进行大量的硬件检测和初始化工作，而嵌入式的硬件都是固定的，只需要选择需要的硬件驱动就可以，不需要全部的硬件驱动都检测；因此可以进行适当的裁剪内核达到缩小启动 linux 系统的目的；同时可以统计驱动模块的耗时时间，对耗时较长的模块驱动加以分析，优化

309.【简答题】【体系内核与基础】简述 PSRAM、SDRAM、DDR、DDR2 的时序特性?

PSRAM，全称 Pseudo static random access memory。指的是伪静态随机存储器。

SDRAM:Synchronous Dynamic Random Access Memory，同步动态随机存储器，同步是指 Memory 工作需要同步时钟，内部的命令的发送与数据的传输都以它为基准;动态是指存储阵列需要不断的刷新来保证数据不丢失;随机是指数据不是线性依次存储，而是自由指定地址进行数据读写。

DDR=Double Data Rate 双倍速率同步动态随机存储器。DDR SDRAM 是 Double Data Rate SDRAM 的缩写，是

双倍速率同步动态随机存储器的意思。

在同等核心频率下，DDR2 的实际工作频率是 DDR 的两倍。这得益于 DDR2 内存拥有两倍于标准 DDR 内存的 4BIT 预读取能力。换句话说，虽然 DDR2 和 DDR 一样，都采用 DDR2 内存的频率了在时钟的上升延和下降延同时进行数据传输的基本方式，但 DDR2 拥有两倍于 DDR 的预读取系统命令数据的能力。也就是说，在同样 100MHz 的工作频率下，DDR 的实际频率为 200MHz，而 DDR2 则可以达到 400MHz

310.【简答题】【体系与内核基础】什么是中断?中断发生时 CPU 做什么工作?

中断是指来自 CPU 执行指令以外的事件发生后，处理机暂停正在运行的程序，转去执行处理该事件的程序的过程，需要 cpu 暂停当前的任务，做相应的处理，cpu 需要判断中断源，保存现场状态，以便能够处理完后继续执行中断的任务。

311.【开放题】【体系与内核基础】请选出你使用过的嵌入式处理器类型：

□ARM7□Cortex □ARM9□ARM11
□MIPS□Power PC□DSP□NIOSII
□MicroBlaze □MCS51 系列单片机□AVR 系列单片机
□其他类型的嵌入式处理器

312.【简答题】【体系内核与基础】嵌入式处理器（ARM）中 IRQ 和 FTQ 有什么区别，在 CPU 里面是怎么实现的

ARM 的 FIQ 模式提供了更多的 banked 寄存器，r8 到 r14 还有 SPSR，而 IRQ 模式就没有那么多，R8,R9,R10,R11,R12 对应的 banked 的寄存器就没有，这就意味着在 ARM 的 IRQ 模式下，中断处理程序自己要保存 R8 到 R12 这几个寄存器，然后退出中断处理时程序要恢复这几个寄存器，而 FIQ 模式由于这几个寄存器都有 banked 寄存器，模式切换时 CPU 自动保存这些值到 banked 寄存器，退出 FIQ 模式时自动恢复，所以这个过程 FIQ 比 IRQ 快.

FIQ 比 IRQ 有更高优先级，如果 FIQ 和 IRQ 同时产生，那么 FIQ 先处理。

在 symbian 系统里，当 CPU 处于 FIQ 模式处理 FIQ 中断的过程中，预取指令异常，未定义指令异常，软件中断全被禁止，所有的中断被屏蔽。所以 FIQ 就会很快执行，不会被其他异常或者中断打断，所以它又比 IRQ 快了。而 IRQ 不一样，当 ARM 处理 IRQ 模式处理 IRQ 中断时，如果来了一个 FIQ 中断请求，那正在执行的 IRQ 中断处理程序会被抢断，ARM 切换到 FIQ 模式去执行这个 FIQ，所以 FIQ 比 IRQ 快多了。

另外 FIQ 的入口地址是 0x1c,IRQ 的入口地址是 0x18

313.【单选题】【操作系统】关于线程和进程的区别，下列描述错误的是（C）

A．线程作为调度和分配的基本单位，进程作为拥有资源的基本单位

B．进行之间可以并发执行，多个线程也可以并发执行

C．进程是拥有资源的独立单位，线程也是拥有资源的独立单位

D．线程可以访问隶属于进程的资源

314.【单选题】【操作系统】下列关于存储单元(MMU)说法错误的是（B）

A、MMU 提供一个关键服务是使各个任务作为各个独立的程序在弃自己私有存储空间中运行

B、在带 MMU 的操作系统的控制下，运行的任务必须知道其他与之无关的任务的存储需求情况，这就简化了各个任务的设计

C、MMU 提供一些资料员以允许使用虚拟存储器

D、MMU 作为转换器，将程序和数据的虚拟地址(编译是的连接地址)转换成实际的物理地址，即在物理主存中的地址

315.【开放题】【接口与应用】 请选出你在产品设计中使用过的串行数字通讯接口

☐Ethernet ☐USB

☐3G (mobi le)☐GPRS GMS ☐ 其它

☐1EEE1394 ☐RS232

☐CAN

☐RS485/422

316.【单选题】【计算机组成原理】存储一个 32 位数 0x12345678 到 2000H ~ 2003H 四个字节的单元中，若以大端模式存储，则 2000H 存储单元的内容为（A）

A、0x21 B、0x68 C、 0x65 D、0x02

317.【单选题】【GCC 编程】GCC 编译时有如下命令：GCC -g test.c -o test，其中参数 g 的作用是（D）

A、生成目标文件 test.o B、生成汇编文件 test.s

C、进行预编译 D、包含调试信息

318.【单选题】【计算机网络】属于网络层协议的是：(B)

A、 TCP　　B、 IP　　C、UDP　　D、 X.25

319.【单选题】【计算机网络】Internet 物理地址和 IP 地址转换成应用的协议是（C）

A、 IP　　B、 TCP　　C、ARP　　D、IGMP

320.【单选题】【计算机网络】下列哪个不属于网络 7 层模型定义（D）

A、 物理层　　B、 链路层　　C、 网络层　　D、 设备层

321.【单选题】【计算机网络】E 类网络 IP 地址的起始地址前 8 位是（B）

A、 192　　B、 224　　C、 172　　D、 238

322.【填空题】【计算机网络】设置一绝对地址为 0x67a9 的整形变量的值为 0xaa66_____

_0xaa55

323.【单选题】【计算机网络】IPv6 地址长度是（C）位

A、 32　　B、 64　　C、 128　　D、 256

324.【简答题】【私有云】私有云的优点有哪些？

数据安全、SLA(服务质量) 、充分利用现有硬件资源和软件资源、不影响现有 IT 管理的流程

325.【单选题】【公有云】公有云计算基础架构的基石是（）

A、虚拟化
B、分布式
C、并行
D、集中式

答案：B

解析：因为公有云计算基础架构的基石是分布式。。

326.【简答题】【公有云】公有云的核心属性是什么？

共享资源服务。

327.【多选题】【公有云】讯飞语音识别公有云计算模型分为哪三个部分？（）

A、公有云接入

B、公有云平台

C、公有云管理

D、公有云维护

答案：ABC

解析：公有云维护不属于计算模型的三个部分。

328.【简单题】【混合云】请简单叙述混合云：

混合云是云计算的一种类型，它将本地基础结构（或私有云）与公有云结合在一起。使用混合云，可以在两种环境之间移动数据和应用。

329.【简答题】【混合云】请简述混合云平台的优势

1 、混合云平台为组织提供了许多优势，例如更大的灵活性、更多的部署选项、更高的安全性和符合性，并能从其现有基础结构获得更多价值。

2 、当计算和处理需求发生变化时，混合云计算使企业能够将其本地基础结构无缝扩展到公有云以处理任何溢出，而无需授予第三方数据中心访问其完整数据的权限。

3、通过在云中运行特定工作负载，组织可以获得公有云提供的灵活性和创新能力，同时将高度敏感的数据保存在自己的数据中心内，满足客户需求或监管要求。

330.【多选题】【混合云服务】以下构建私有云的步骤有哪些（）

A、云计算

B、应用管理

C、业务进程

D、架构影响

答案：A、B、C、D

解析：云是用来根据用户需求将 IT 和技术转化为服务的一系列规则、技术及业务模式。系统管理程序肯定是需要考虑的一层，提供内部云服务的技术已经出现，而且在逐渐走向成熟。但是进程排列、资产管理、安全政策制定、进程支持及统计等业务依然需要大量的工作。

331.【单选题】【混合云服务】以下不是实现云计算的解决方案是（）

A、渐进式

B、演进式

C、革命性

D、转变策略

A：两条路实现云计算的解决方案中不包含 渐进式。

332.【多选题】【混合云服务】讯飞语音识别公有云提供了哪些内容（）

A、语音识别

B、语音合成

C、语音拓展

D、文字识别

答案：ABCD

解析：面前讯飞语音识别公有云上提供了语音识别、合成、拓展、人脸识别、文字识别、图形识别等诸多资源。

333.【多选题】【混合云服务】讯飞语音识别公有云提供了哪些系统环境下的 SDK 文档内容（）

A、Android SDK

B、IOS SDK

C、LINUX SDK

D、WINDOWS SDK

答案：ABCD

解析：面前讯飞语音识别公有云上提供了 Android SDK 、IOS SDK 、LINUX SDK、 WINDOWS SDK

334.【多选题】【混合云服务】腾讯人脸识别公有云环境下的离线识别 SDK 应用场景有哪些？（）

A、门禁考勤

B、人证核验

C、入口闸机

D、智能家居

答案：ABCD

解析：离线识别 SDK 应用场景如下：

门禁考勤：公司、学校、小区、楼宇等。

人证核验：酒店、机场/车站、政务业务等。

入口闸机：景区、园区、校园、工地等。

自助设备：零售、银行、医院、政务大厅等。

机器人：家庭、教育、服务等机器人。

智能家居：智能空调、智能电视、智能冰箱等。

335.【简答题】【混合云服务】请简述腾讯人脸识别的离线识别 SDK 的子集和相关功能？

1、人脸跟踪（YTFaceTracker）：快速检测出画面中的人脸，获得【人脸大小、位置、角度】，并由此可以进行初步筛选。

2、人脸精确配准（YTFaceAlignment）：获得人脸关键点坐标，以及关键点可见度，用于遮挡判断，表情判断。（在戴口罩的场景下，由于脸部特征缺失，不建议使用此能力。）

3、人脸质量（YTFaceQuality）：人脸图片质量打分，用于照片优选。

4、人脸质量归因（YTFaceQualityPro）：获得人脸的【角度、遮挡、模糊、光照】几个维度的分析结果， 用于进一步照片筛选。（在戴口罩的场景下，由于脸部特征缺失，不建议使用此能力。）

5、彩色图活体（YTFaceLiveColor）：仅通过普通单目摄像头输出的彩色图判断当前人脸是否为活体（防止照片、显示屏等翻拍攻击）。

6、红外活体（YTFaceLiveIR）：红外摄像头获得的图像与彩色摄像头获得的图像一起分析，判断摄像头前面的人是否为活体。

7、深度活体（YTFaceLive3D）：深度摄像头获得的图像与彩色摄像头获得的图像一起分析，判断摄像头前面的人是否为活体。

8、人脸提特征（YTFaceFeature）：提取人脸特征，并可进行人脸 1:1 比对（比较 2 个人脸相似度，判断是否同一人），提特征同时也是 1:N 搜索的前提。

336.【单选题】【混合云服务】百度地图公有云采用了可视化的 ___来进行（）

A、Java

B、PHP

C、Sugar BI

D、C#

答案：C

解析：Sugar BI 是百度智能云推出的敏捷 BI 和数据可视化平台，通过拖拽图表组件可实现 5 分钟搭建数据可视化页面，组件丰富，开箱即用，无需 SQL 和任何编码。通过可视化图表及强大的交互分析能力，企业可使用 Sugar BI 有效助力自己的业务决策。

337.【简答题】【混合云服务】请简述百度地图公有云的 Sugar BI 的基本内容。

Sugar BI 是百度云推出的敏捷 BI 和数据可视化平台，目标是解决报表和大屏的数据 BI 分析和可视化问题，解放数据可视化系统的开发人力。Sugar BI 提供界面优美、体验良好的交互设计，通过拖拽图表组件可实现 5 分钟搭建数据可视化页面，并对数据进行快速的分析。通过可视化图表及强大的交互分析能力，企业可使用 Sugar BI 有效助力自己的业务决策。

338.【简答题】【混合云服务】请简述公有云存在的意思

能够以低廉的价格，提供有吸引力的服务给最终用户，创造新的业务价值，公有云作为一个支撑平台，还能够整合上游的服务（如增值业务，广告）提供者和下游最终用户，打造新的价值链和生态系统。

339.【简答题】【混合云服务】请简述公有云存在的意思

能够以低廉的价格，提供有吸引力的服务给最终用户，创造新的业务价值，公有云作为一个支撑平台，还能够整合上游的服务（如增值业务，广告）提供者和下游最终用户，打造新的价值链和生态系统。

340.【简答题】【混合云服务】请简述百度地图公有云存在的特点

1. 面向 web 应用和移动应用

2. 将端和云整合在一起。尽管百度没有自己的应用商店，但是百度最近收购了 91，其企图是通过强化移动入口吸引开发者，这样，百度就有了两个入口（搜索和手机助手）。因此，开发者门户并不是单纯的云服务门户，而是云+端+一系列工具一体的服务平台。

341.【多选题】【混合云服务】阿里通讯公有云物联网平台一般是哪几部分组成（）

A、设备通过物模型上报数据属性

B、App 通过云端 API 获取设备的数据属性

C、App 通过云端 API 调用设备的服务，来控制设备执行各种命令

D、通过官方 API 直接调用

答案：ABC

解析：因为 D 不属于阿里云物联网平台

342.【多选题】【混合云服务】阿里通信一键登录的系统交互流程主要分为哪几步骤？（）

A、号码认证 SDK 初始化

B、唤起授权页

C、同意授权并登录

D、发起取号

答案：ABCD,

解析：开发者需要在 APP 中集成号码认证服务客户端 SDK，并在服务端完成 API 对接。

1、一键登录的系统交互流程主要分为四个步骤：

第一步，号码认证 SDK 初始化

第二步，唤起授权页

第三步，同意授权并登录

第四步，发起取号

343.【简答题】【混合云服务】请简述公有云与私有云的区别

1、用户上： 公有云：创业公司、个人 ； 私有云：政府、大企业

2、业务场景：公有云：对外互联网业务 私有云：政府企业内部业务

3、技术架构：公有云：自研架构、关注分布式、大集群 私有云： OpenStack 开源架构关注高可用、灵活性

4、安全性：公有云：主机层实现安全隔离 ；私有云：网络层 实现安全隔离

344.【多选题】【混合云服务】云的架构（服务模式）有哪些（）

A、IaaS

B、PaaS

C、SaaS

D、FaaS

答案：D，

解析：

IaaS:Infrastrure-as-Service（基础设施即服务）

PaaS：platform-as-a-Service（平台即服务）

SaaS：Software-as-a-Service（软件即服务）

345.【多选题】【混合云服务】云的部署方式有哪些（）

A、公有云

B、私有云

C、社区云

D、混合云

答案：ABCD

解析：四种部署方式（公有云，私有云，社区云，混合云）

346.【单选题】【混合云服务】讯飞语音识别云服务器是什么（）

A、CVM

B、DBS

C、DOTA

D、LOL

答案：A

解析：云服务器 CVM 是科大讯飞提供的一种可扩展的基础云计算服务

347.【多选题】【混合云服务】讯飞语音识别云服务器的优势 有哪些？（）

A、弹性

B、安全

C、极速

D、轻便

答案：ABC:

解析：多种方式远程登录云服务器：提供多种登录方式，包括密码登录、控制台登录等。

无论从用户操作还是云服务器性能，都致力于提供极速便捷的服务。

操作便捷快速：您只需几分钟时间即可轻松获取一个、数百个服务器实例，您可以一键购买、配置、扩展、管理您的服务

348.【单选题】【混合云服务】讯飞语音识别云的公共镜像的特质有哪些？（）

A、操作系统类型

B、安全

C、限制

D、费用

答案：ABCD

解析：公共镜像是由云服务器 CVM 提供、支持和维护的镜像，包含基础操作系统和讯飞云提供的初始化组件，所有用户均可使用。 公共镜像特质：

操作系统类型：自由选择（如：基于 Linux 系统的多种镜像），并定期更新。

安全：提供的操作系统完全合法合规，均使用官方正版操作系统。

限制：暂无使用限制。

费用：镜像使用全部免费。

349.【简答题】【嵌入式系统】bootloader 在嵌入式系统中主要起什么作用？完成哪些主要的工作？

在嵌入式操作系统中，BootLoader 是在操作系统内核运行之前运行。可以初始化硬件设备、建立内存空间映射图，从而将系统的软硬件环境带到一个合适状态，以便为最终调用操作系统内核准备好正确的环境。在嵌入式系统中，通常并没有像 BIOS 那样的固件程序（注，有的嵌入式 CPU 也会内嵌一段短小的启动程序），因此整个系统的加载启动任务就完全由 BootLoader 来完成。

350.【填空题】【嵌入式系统】嵌入式操作系统主要有_____。

DOS、Windows CE、Palm、Linux

351.【单选题】【嵌入式系统】嵌入式系统有硬件和软件部分构成，以下（A）不属于嵌入式软件。

A．系统软件　　B、驱动　　C、ADS 软件　　D、嵌入式中间件

352.【单选题】【嵌入式系统】实时操作系统中，两个任务并发执行，一个任务要等待其他合作伙伴发来消息，或者建立某个条件后再向前执行，这种制约性合作关系被称为（B）。

A、 同步　　B、 互斥　　C、 调度　　D、 执行

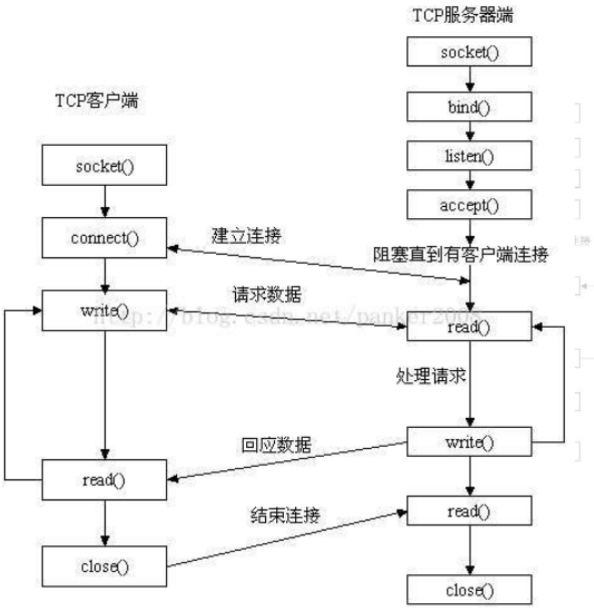
353.【单选题】【嵌入式系统】下面那点不是嵌入式操作系统的特点（C）

A、内核精简　　B、专用性强　　C、功能强大　　D、高实时性

354.【单选题】【嵌入式系统】嵌入式系统的开发通常是在交叉开发环境实现的，交叉开发环境指的是（A）

A、在宿主机上开发，在目标机上运行
B、在目标机上开发，在宿主机上运行
C、在宿主机上开发， 在宿主机上运行
D、在目标机上开发，在目标机上运行

355.【简答题】【嵌入式系统】Socket 编程中，tcp 服务器端与 tcp 客户端的步骤分别是什么？



356.【简答题】【嵌入式系统】简述 ksoftirqd 内核线程

内核不会立即处理重新触发的软中断。当大量软中断出现的时候，内核会唤醒一组内核线程来处理。这些线程的优先级最低(nice 值为 19)，这能避免它们跟其它重要的任务抢夺资源。但它们最终肯定会被执行，所以这个折中的方案能够保证在软中断很多时用户程序不会因为得不到处理时间而处于饥饿状态，同时也保证过量的软中断最终会得到处理。

每个处理器都有一个这样的线程，名字为 ksoftirqd/n，n 为处理器的编号。

357.【简答题】【嵌入式系统】简述嵌入式系统的定义、应用和特点?

嵌入式系统定义：以应用为中心，以计算机技术为基础，其软硬件可裁剪配置，对功能、可靠性、成本、体积、功耗有严格约束的一种专用计算机系统。

嵌入式系统应用：应用于军事设备、信息终端、汽车电子、制造工业、航天航空等领域。

嵌入式系统特点：专用性、可裁性、实时性好、可靠性高、功耗低。

358.【简答题】【嵌入式系统】管道和套接字有什么区别？ 分别适用于哪些场景?

管道：管道的本质其实就是内核中的一块缓冲区，多个进程通过访问同一个缓冲区就可以实现进程间的通信

Socket：是对网络中不同主机上的应用进程之间进行双向通信的端点的抽象。一个套接字就是网络上进程通信的一端，提供了应用层进程利用网络协议交换数据的机制。完成两个应用程序之间的数据传输。

对于 TCP/IP 套接字，数据传输的效率更高，开销也更少。数据传输还可以利用 TCP/IP 套接字性能增强机制的优点，如开窗口、延迟确认等，这在慢速网络中可能非常有益。对于应用程序的不同类型，这类性能差异可能非常大。

TCP/IP 套接字还支持待办事项队列，当试图连接到 SQL Server 时，与可能导致管道忙错误的命名管道相比，该队列可以提供有限的平稳效果。

359.【简答题】【嵌入式系统机制】产生死锁的原因是什么？

多个并发进程因争夺系统资源而产生相互等待的现象。即：一组进程中的每个进程都在等待某个事件发生，而只有这组进程中的其他进程才能触发该事件，这就称这组进程发生了死锁。

产生死锁的本质原因为：

1)、系统资源有限。

2)、进程推进顺序不合理。

360.【简答题】【嵌入式系统机制】死锁的 4 个必要条件

1、互斥：某种资源一次只允许一个进程访问，即该资源一旦分配给某个进程，其他进程就不能再访问，直到该进程访问结束。

2、占有且等待：一个进程本身占有资源（一种或多种），同时还有资源未得到满足，正在等待其他进程释放该资源。

3、不可抢占：别人已经占有了某项资源，你不能因为自己也需要该资源，就去把别人的资源抢过来。

4、循环等待：存在一个进程链，使得每个进程都占有下一个进程所需的至少一种资源。

当以上四个条件均满足，必然会造成死锁，发生死锁的进程无法进行下去，它们所持有的资源也无法释放。这样会导致 CPU 的吞吐量下降。所以死锁情况是会浪费系统资源和影响计算机的使用性能的。那么，解决死锁问题就是相当有必要的了。

361.【简答题】【嵌入式系统机制】死锁的处理方式有哪些？

死锁的处理方式主要从预防死锁、避免死锁、检测与解除死锁这四个方面来进行处理。

预防死锁：

1、资源一次性分配：（破坏请求和保持条件）

2、可剥夺资源：即当某进程新的资源未满足时，释放已占有的资源（破坏不可剥夺条件）

3、资源有序分配法：系统给每类资源赋予一个编号，每一个进程按编号递增的顺序请求资源，释放则相反（破坏环路等待条件）

避免死锁:

预防死锁的几种策略，会严重地损害系统性能。因此在避免死锁时，要施加较弱的限制，从而获得 较满意的系统性能。由于在避免死锁的策略中，允许进程动态地申请资源。因而，系统在进行资源分配之前预先计算资源分配的安全性。若此次分配不会导致系统进入不安全状态，则将资源分配给进程；否则，进程等待。其中最具有代表性的避免死锁算法是银行家算法。

检测死锁：

首先为每个进程和每个资源指定一个唯一的号码；

然后建立资源分配表和进程等待表

362.【简答题】【嵌入式系统机制】当发现有进程死锁后，便应立即把它从死锁状态中解脱出来，常采用的方法有：

1、剥夺资源：从其它进程剥夺足够数量的资源给死锁进程，以解除死锁状态；

2、撤消进程：可以直接撤消死锁进程或撤消代价最小的进程，直至有足够的资源可用，死锁状态.消除为止；所谓代价是指优先级、运行代价、进程的重要性和价值等。

363.【简答题】【嵌入式系统机制】进程和线程有什么区别？

进程是并发执行的程序在执行过程中分配和管理资源的基本单位。线程是进程的一个执行单元，是比进程还要小的独立运行的基本单位。一个程序至少有一个进程，一个进程至少有一个线程。两者的区别主要有以下几个方面：

1. 进程是资源分配的最小单位。

2. 线程是程序执行的最小单位，也是处理器调度的基本单位，但进程不是，两者均可并发执行。

3. 进程有自己的独立地址空间，每启动一个进程，系统就会为它分配地址空间，建立数据表来维护代码段、堆栈段和数据段，这种操作非常昂贵。而线程是共享进程中的数据，使用相同的地址空间，因此，CPU 切换一个线程的花费远比进程小很多，同时创建一个线程的开销也比进程小很多。

4. 线程之间的通信更方便，同一进程下的线程共享全局变量、静态变量等数据，而进程之间的通信需要以通信的方式（IPC)进行。不过如何处理好同步与互斥是编写多线程程序的难点。但是多进程程序更健壮，多线程程序只要有一个线程死掉，整个进程也跟着死掉了，而一个进程死掉并不会对另外一个进程造成影响，因为进程有自己独立的地址空间。

5. 进程切换时，消耗的资源大，效率低。所以涉及到频繁的切换时，使用线程要好于进程。同样如果要求同时进行并且又要共享某些变量的并发操作，只能用线程不能用进程。

6. 执行过程：每个独立的进程有一个程序运行的入口、顺序执行序列和程序入口。但是线程不能独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。

364.【简答题】【嵌入式系统机制】对比进程和线程的优缺点：

线程执行开销小，但是不利于资源的管理和保护。线程适合在 SMP 机器（双 CPU 系统）上运行。

进程执行开销大，但是能够很好的进行资源管理和保护，可以跨机器迁移。

何时使用多进程，何时使用多线程？

对资源的管理和保护要求高，不限制开销和效率时，使用多进程。

要求效率高，频繁切换时，资源的保护管理要求不是很高时，使用多线程。

365.【简答题】【嵌入式系统机制】列举几种进程的同步机制，并比较其优缺点。

原子操作 信号量机制 自旋锁 管程，会合，分布式系统

366.【简答题】【嵌入式系统机制】产生死锁的原因是什么？

多个并发进程因争夺系统资源而产生相互等待的现象。即：一组进程中的每个进程都在等待某个事件发生，而只有这组进程中的其他进程才能触发该事件，这就称这组进程发生了死锁。

产生死锁的本质原因为：

1)、系统资源有限。

2）、进程推进顺序不合理。

367.【简答题】【嵌入式系统机制】CSingleLock 是干什么的。

同步多个线程对一个数据类的同时访问

368.【简答题】【嵌入式系统机制】Linux 有内核级线程么。

线程通常被定义为一个进程中代码的不同执行路线。从实现方式上划分，线程有两种类型：“用户级线程”和“内核级线程”。 用户线程指不需要内核支持而在用户程序中实现的线程，其不依赖于操作系统核心，应用进程利用线程库提供创建、同步、调度和管理线程的函数来控制用户线程。这种线程甚至在象 DOS 这样的操作系统中也可实现，但线程的调度需要用户程序完成，这有些类似 Windows 3.x 的协作式多任务。另外一种则需要内核的参与，由内核完成线程的调度。其依赖于操作系统核心，由内核的内部需求进行创建和撤销，这两种模型各有其好处和缺点。用户线程不需要额外的内核开支，并且用户态线程的实现方式可以被定制或修改以适应特殊应用的要求，但是当一个线程因 I/O 而处于等待状态时，整个进程就会被调度程序切换为等待状态，其他线程得不到运行的机会；而内核线程则没有各个限制，有利于发挥多处理器的并发优势，但却占用了更多的系统开支。

Windows NT 和 OS/2 支持内核线程。Linux 支持内核级的多线程

369.【简答题】【嵌入式系统机制】使用线程是如何防止出现大的波峰。

意思是如何防止同时产生大量的线程，方法是使用线程池，线程池具有可以同时提

高调度效率和限制资源使用的好处，线程池中的线程达到最大数时，其他线程就会排队

等候

370.【简答题】【嵌入式系统机制】 嵌入式系统总是要用户对变量或寄存器进行位操作。给定一个整型变量 a，写两段代码，第一个设置 a 的 bit 3，第二个清除 a 的 bit 3。在以上两个操作中，要保持其它位不变。对这个问题有三种基本的反应

1). 不知道如何下手。该被面者从没做过任何嵌入式系统的工作。

2). 用 bit fields。Bit fields 是被扔到 C 语言死角的东西，它保证你的代码在不同编译器之间是不可移植的，同时也保证了的你的代码是不可重用的。我最近不幸看到 Infineon 为其较复杂的通信芯片写的驱动程序，它用到了 bit fields 因此完全对我无用，因为我的编译器用其它的方式来实现 bit fields 的。从道德讲：永远不要让一个非嵌入式的家伙粘实际硬件的边。

371.【简答题】【嵌入式系统机制】用户空间与内核通信方式有哪些？

1)系统调用。用户空间进程通过系统调用进入内核空间，访问指定的内核空间数据；

2)驱动程序。用户空间进程可以使用封装后的系统调用接口访问驱动设备节点，以和运行在内核空间的驱动程序通信；

3)共享内存 mmap。在代码中调用接口，实现内核空间与用户空间的地址映射，在实时性要求很高的项目中为首选，省去拷贝数据的时间等资源，但缺点是不好控制；

4)copy_to_user()、copy_from_user()，是在驱动程序中调用接口，实现用户空间与内核空间的数据拷贝操作，应用于实时性要求不高的项目中。

372.【简答题】【嵌入式系统机制】MMU 基础

现代操作系统普遍采用虚拟内存管理（Virtual Memory Management） 机制，这需要 MMU（ Memory Management Unit，内存管理单元） 的支持。有些嵌入式处理器没有 MMU，则不能运行依赖于虚拟内存管理的操作系统。

也就是说：操作系统可以分成两类，用 MMU 的、不用 MMU 的。

用 MMU 的是：Windows、MacOS、Linux、Android；不用 MMU 的是：FreeRTOS、VxWorks、UCOS……

与此相对应的：CPU 也可以分成两类，带 MMU 的、不带 MMU 的。

带 MMU 的是：Cortex-A 系列、ARM9、ARM11 系列；

不带 MMU 的是：Cortex-M 系列……（STM32 是 M 系列，没有 MMU，不能运行 Linux，只能运行一些 UCOS、FreeRTOS 等等）。

MMU 就是负责虚拟地址（virtual address）转化成物理地址（physical address），转换过程比较复杂，可以自行百度。

373.【编程题】【嵌入式系统库函数实现】不使用库函数，编写函数 int strcmp(char *source, char *dest) 相等返回 0，不等返回-1；

一、

```
intstrcmp(char   *source, char *dest)
 {
 assert((source!=NULL)&&(dest!=NULL));
int i,j;
for(i=0;source[i]==dest[i]; i++)
 {
 if(source[i]=='\0' &&dest[i]=='\0')
return 0;  else    return-1;
```

```
 }
 }
```

二、

```
intstrcmp(char *source, char *dest)
{
while ( (*source != '\0')&& (*source == *dest))
{
source++;dest++;
}
return ( (*source) - (*dest) ) ?-1 : 0;
}
```

374.【解答题】【嵌入式产品设计】

有一个使用 UART 进行通信的子系统 X，其中 UART0 进行数据包接收和回复，UART1 进行数据包转发。子系统 X 的通信模块职责是从 RART0 接收数据包，如果为本地数据包（reveive 为子系统 X），则解析数据包中的命令码（2 字节）和数据域（0-127 字节），根据命令码调用内部的处理程序，并将处理结果通过 UART0 回复给发送端，如非本地数据包，则通过 UART1 转发。

如果由你来设计子系统 X 的通信模块：

1）请设计通信数据包格式，并说明各字段的定义：（总分 5 分）

2）在一个实时操作系统中，你会如何部署模块中的任务和缓存数据（总分 5 分）

3）你会如何设置任务的优先级，说说优缺点：（总分 5 分）

4）如何将命令码对应的处理优先级分为高、低两个等级，你又会如何设计。（总分 5 分）

（1）添加一个字节的标识字节。第一位用来区分是本地数据包还是，非本地数据包。后 7 位用来表示数据域字节长度。

（2）为串口的接收和发送分别创建 2 个数据缓冲区，并创建两个缓冲区管理任务，一个用来接收数据时，阻塞任何访问缓冲区数据的任务。发送缓冲区为 UART0 和 UART1 共享的，当接收的到缓冲命令时，阻塞任何向发送缓冲区写数据的操作任务。

（3）任务优先级为，接收的优先级大于发送的优先级。应为如果不及时接收可能造成数据包的丢失

（4）优先级继承：当高优先级的任务需要等待低优先级任务释放互斥资源时，暂时先把低优先级任务的优先级提升至和高优先级任务优先级一样。这样就确保了不会有其他中优先级任务抢占执行。

极限优先级：当任务占用互斥资源执行临界区代码时，先把该任务的优先级提升至极限优先级（系统最高优先级），等到释放资源时再降回原有优先级。

375.【代码分析题】【嵌入式程序设计】

```
#include   <stdliB、h>
#include <stdio.h>
void getmemory(char *p)
{
p=(char *)malloc(100);
strcpy(p,"helloworld");
}
int main( )
{
char *str=NULL;
getmemory(str);
printf("%s/n",str);
free(str);
return 0;
}
```

程序崩溃，getmemory 中的 malloc 不能返回动态内存， free()对 str 操作很危险

376.【代码分析题】【嵌入式系统中断】 中断是嵌入式系统中重要的组成部分，这导致了很多编译开发商提供一种扩展—让标准 C 支持中断。具代表事实是，产生了一个新的关键字 __interrupt。下面的代码就使用了__interrupt 关键字去定义了一个中断服务子程序(ISR)，请评论一下这段代码的。

```
__interrupt double compute_area (double radius)
{
double area = PI * radius * radius;
printf(" Area = %f", area);
return area;
}
```

1). ISR 不能返回一个值。如果你不懂这个，那么你不会被雇用的。

2). ISR 不能传递参数。如果你没有看到这一点，你被雇用的机会等同第一项。

3). 在许多的处理器/编译器中，浮点一般都是不可重入的。有些处理器/编译器需要让额处的寄存器入栈，有些

处理器/编译器就是不允许在 ISR 中做浮点运算。此外，ISR 应该是短而有效率的，在 ISR 中做浮点运算是不明智的。

4). 与第三点一脉相承，printf()经常有重入和性能上的问题。如果你丢掉了第三和第四点，我不会太为难你的。不用说，如果你能得到后两点，那么你的被雇用前景越来越光明了。

377.【简答题】【嵌入式系统内核】系统调用与普通函数调用的区别

系统调用：

1.使用 INT 和 IRET 指令，内核和应用程序使用的是不同的堆栈，因此存在堆栈的切换，从用户态切换到内核态，从而可以使用特权指令操控设备

2.依赖于内核，不保证移植性

3.在用户空间和内核上下文环境间切换，开销较大

4.是操作系统的一个入口点

普通函数调用：

1.使用 CALL 和 RET 指令，调用时没有堆栈切换

2.平台移植性好

3.属于过程调用，调用开销较小

4.一个普通功能函数的调用

378.【简答题】【嵌入式系统分类】列举出几个硬实时系统和软实时系统

软实时系统：

Windows、Linux 系统通常为软实时，当然有补丁可以将内核做成硬实时的系统，不过商用没有这么做的。

硬实时系统：

对时间要求很高，限定时间内不管做没做完必须返回。

VxWorks，uCOS，FreeRTOS，WinCE，RT-thread 等实时系统；

379.【简答题】【嵌入式 linux 系统分析】 Linux 操作系统挂起、休眠、关机相关命令

关机命令有 halt， init 0， poweroff ，shutdown -h 时间，其中 shutdown 是最安全的

重启命令有 reboot，init 6,，shutdow -r 时间

在 linux 命令中 reboot 是重新启动，shutdown -r now 是立即停止然后重新启动

具体可用参数可以百度。

380.【简答题】【makefile】针对源文件 test.c， 编写一个 Makefile。

A、out:test.c

```
    gcc -o A、out test.c
clean:
    rm -f A、out
.PHONY: clean
```