腾讯2013研发工程师笔试题

一. 单项选择题

1. 下面的排序算法中，初始数据集的排列顺序对算法的性能无影响的是

A 插入排序
B 堆排序
C 冒泡排序
D 快速排序

2. 以下表的设计，最合理的是（ ）

A 学生{id,name,age} ,学科{id,name} 分数{学生 id,学科 id,分数}
B 学生{id,name,age} ,分数{学生 id,学科名称,分数}
C 分数{学生姓名,学科名称,分数}
D 学科{id,name},分数{学生姓名,学科 id,分数}

3. 在数据库系统中，产生不一致的根本原因是（ ）

A 数据存储量太大
B 没有严格保护数据
C 未对数据进行完整性控制
D .数据冗余

4. 用容积分别为15升和27升的两个杯子向一个水桶中装水，可以精确向水桶中注入（ ）升水?

A 53
B 25
C 33
D 52

5. 考虑左递归文法S->Aalb A->AclSdle，消除左递归后应该为（ ）？

A S->Aalb A->bdA'lA' A'->cA'ladA'le
B S->Abla A->bdA'lA' A'->cA'ladA'le
C S->Aalb A->cdA'lA' A'->bA'ladA'le
D S->Aalb A->bdA'lA' A'->caA'ldA'le

6. 使用二分查找算法在一个有序序列中查找一个元素的时间复杂度为（ ）

A O(N)
B O(logN)
C O(N*N)
D O(N*logN)

7. 路由器工作在网络模型中的哪一层（）？

A 数据链路层

B 物理层

C 网络层

D 应用层

8. 对于满足SQL92标准的SQL语句：SELECT foo,count(foo) FROM pokes WHERE foo>10 GROUP BY foo HAVINO ORDER BY foo，其执行的顺序应该为（）

A FROM->WHERE->GROUP BY->HAVING->SELECT->ORDER BY

B FROM->GROUP BY->WHERE->HAVING->SELECT->ORDER BY

C FROM->WHERE->GROUP BY->HAVING->ORDER BY->SELECT

D FROM->WHERE->ORDER BY->GROUP BY->HAVING->SELECT

9. 在UNIX系统中，目录结构采用（）

A 单级目录结构

B 二级目录结构

C 单纯树形目录结构

D 带链接树形目录结构

10.
请问下面的程序一共输出多少个“-”?

```
int main(void)
{
    int i;
    for (i = 0; i < 2; i++) {
        fork();
        printf("-");
    }
    return 0;
}
```

A 2

B 4

C 6

D 8

11.
// 请问下面的程序一共输出多少个“-”? 为什么?

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

```
#include <sys/types.h>
#include <unistd.h>

int main(void) {
   int i;
   for (i=0; i<2; i++) {
      fork();
      printf("-\n");
   }
   return 0;
}
```

A 4

B 5

C 6

D 8

12. 避免死锁的一个著名的算法是（ ）

A 先入先出法

B 银行家算法

C 优秀级算法

D 资源按序分配法

13. 你怎么理解的分配延迟（dispatch lantency）

A 分配器停止一个进程到开启另一个进程的时间

B 处理器将一个文件写入磁盘的时间

C 所有处理器占用的时间

D 以上都不对

14. 以下那一个不是进程的基本状态（ ）

A 阻塞态

B 执行态

C 就绪态

D 完成态

15. 假定我们有3个程序，每个程序花费80%的时间进行I/O，20%的时间使用CPU。每个程序启动时间和其需要使用进行计算的分钟数如下，不考虑进程切换时间：
程序编号 启动时间 需要CPU时间（分钟）
1 00：00 3.5
2 00：10 2
3 00：15 1.5
请问，在多线程/进程环境下，系统的总响应时间为（ ）

A 22.5

B 23.5

C 24.5

D 25.5

16. 在所有非抢占CPU调度算法中，系统平均响应时间最优的是（ ）

A 实时调度算法

B 短任务优先算法

C 时间片轮转算法

D 先来先服务算法

17. 什么是内存抖动（Thrashing）（ ）

A 非常频繁的换页活动

B 非常高的CPU执行活动

C 一个极长的执行进程

D 一个极大的虚拟内存

18. Belady's Anomaly出现在哪里（ ）

A 内存管理算法

B 内存换页算法

C 预防锁死算法

D 磁盘调度算法

19. 以下的生产者消费者程序中，那个不会出现锁死，并且开销最少?
注:
down()
1 判断信号量的取值是否大于等于1
2 如果是，将信号量的值减去一，继续向下执行
3 否则，在该信号量上等待（进城被挂起）
up()
1 将信号量的值增加1（此操作将叫醒一个在信号量上面等待的进程）
2 线程继续往下执行
down()和up()是一组原子操作

A
```
#define N 100 //定义缓冲区的大小
typedef int semaphore; //定义信号量类型
semaphore mutex = 1; //互斥信号量
semaphore empty = N; //缓冲区计数信号量
semaphore full = 0; //缓冲区计数信号量，用来计数缓冲区里的商品数量
void producer(void)
{
    int item;
    while(TRUE){
        item = produce_item();
        down(&empty);
```

```
            down(&empty);
            insert_item(item);
            up(&mutex);
            up(&full);
        }
    }
    void consumer(void)
    {
        int item;
        while(TRUE){
            down(&full);
            down(&mutex);
            item = remove_item();
            up(&mutex);
            up(&empty);
            consume_item(item);
        }
    }
```

B

```
#define N 100
typedef int semaphore;
semaphore empty = N;
semaphore full = 0;
void producer(void)
{
    int item;
    while(TRUE){
        item = produce_item();
        down(&empty);
        insert_item(item);
        up(&full);
    }
}
void consumer(void)
{
    int item;
    while(TRUE){
        down(&full);
        item = remove_item();
        up(&empty);
        consume_item(item);
    }
}
```

C

```
#define N 100
typedef int semaphore;
semaphore mutex = 1;
semaphore empty = N;
semaphore full = 0;
void producer(void)
```

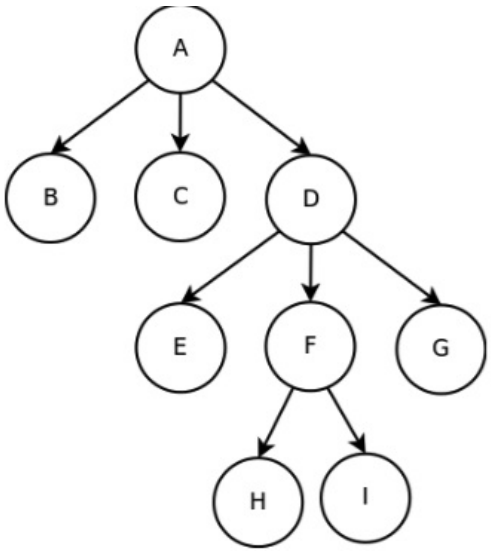
```
{
    int item;
    while(TRUE){
        item = produce_item();
        down(&empty);
        down(&empty);
        insert_item(item);
        up(&mutex);
        up(&full);
    }
}
void consumer(void)
{
    int item;
    while(TRUE){
        down(&mutex);
        down(&full);
        item = remove_item();
        up(&mutex);
        up(&empty);
        consume_item(item);
    }
}
```

D

```
#define N 100
typedef int semaphore;
semaphore mutex = 1;
semaphore empty = N;
semaphore full = 0;
void producer(void)
{
    int item;
    while(TRUE){
        item = produce_item();
        down(&empty);
        down(&mutex);
        insert_item(item);
        up(&full);
        up(&mutex);
    }
}
void consumer(void)
{
    int item;
    while(TRUE){
        down(&full);
        down(&mutex);
        item = remove_item();
        up(&empty);
        up(&mutex);
        consume_item(item);
```

```
        }
    }
```

20.
使用深度有限算法遍历下面的图,遍历的顺序为()



A ABCDEFGHI

B BCEHIFGDA

C ABCDEFHIG

D HIFEGBCDA