



linux 下给文件start.sh 设置权限为自己可读可修改可执行,组内用户为可读可执行不可修改,其余用户没有任何权限,那么设置该文件权限的命令为()

正确答案: B 你的答案: 空 (错误)

```
chmod start.sh 706
chmod start.sh 750
chmod start.sh 705
chmod start.sh 777
```

以下哪种 http 状态下,浏览器会产生两次 http 请求?()

正确答案: C 你的答案: 空 (错误)

304
404
302
400

某学校获取到一个 B 类地址段,要给大家分开子网使用,鉴于现在上网设备急剧增多,管理员给每个网段进行划分的子网掩码设置为 255.255.254.0,考虑每个网段需要有网关设备占用一个地址的情况下,每个网段还有多少可用的主机地址()

正确答案: A 你的答案: 空 (错误)

509
511
512
510

某种产品,合格品率为 0.96,一个合格品被检查成次品的概率是 0.02,一个次品被检查成合格品的概率为 0.05,问题:求一个被检查成合格品的产品确实为合格品的概率()

正确答案: A 你的答案: 空 (错误)

0.9978
0.9991
0.9855
0.96

设某公路上经过的货车与客车的数量之比为 2:1,货车中途停车修理的概率为 0.02,客车为 0.01,今有一辆汽车中途停车修理,求该汽车是货车的概率()

正确答案: D 你的答案: 空 (错误)

0.67
0.33
0.91
0.8



以下多线程对 `int` 型变量 `x` 的操作,哪个不需要进行同步()

正确答案: D 你的答案: 空 (错误)

```
++x
x=y
x++
x=1
```

```
1  #define A(x) x+x
2  int i=5*A(4)*A(6);
3  cout<<i;
```

以上程序输出是多少?

正确答案: A 你的答案: 空 (错误)

50
100
120
480

以下关于 **PMF**(概率质量函数),**PDF**(概率密度函数),**CDF**(累积分布函数)描述错误的是()

正确答案: A 你的答案: 空 (错误)

PDF 描述的是连续型随机变量在特定取值区间的概率

CDF 是 PDF 在特定区间上的积分

PMF 描述的是离散型随机变量在特定取值点的概率

有一个分布的 CDF 函数 $H(x)$, 则 $H(a)$ 等于 $P(X \leq a)$

一个全局变量 `tally`,两个线程并发执行(代码段都是 `ThreadProc`),问两个线程都结束后,`tally` 取值范围为_____

```
1  int tally=0;//全局变量
2      void ThreadProc() {
3          for(int i=1;i<=50;i++)
4              tally+=1;
5      }
```

正确答案: A 你的答案: 空 (错误)

[50,100]
[100,100]
[1275,2550]
[2550,2550]



下面四个类 A,B,C,D,在 32 位机器上 sizeof(A),sizeof(B),sizeof(C),sizeof(D)值分别为()

```
1  class A{
2  };
3  class B{
4      char ch;
5      int x;
6  };
7  class
8  C{ public:
9      void Print(void) {}
10 };
11 class D
12 { public
13 :
14     virtual void Print(void) {}
15 };
```

正确答案: C 你的答案: 空 (错误)

0, 5, 0, 0

1, 8, 4, 4

1, 8, 1, 4

1, 5, 1, 4

下列关于 GIT 的描述不恰当的一项是()

正确答案: C 你的答案: 空 (错误)

可以采用公钥认证进行安全管理

可以利用快照签名回溯历史版本

必须搭建 Server 才能提交修改

属于分布式版本控制工具

已知下面的 class 层次,其中每一个 class 都定义有一个 default constructor 和一个 virtual destructor;

```
1  class X{...};
2  class A{...};
3  class B:public A{...};
4  class C:public B{...};
5  class D:public X, public C{...};
```

下面()执行 dynamic_cast 会失败

正确答案: C 你的答案: 空 (错误)

A *pa=new D;X *px=dynamic_cast<X*>(pa);

D *pd=new D;A *pa=dynamic_cast<A*>(pd);



```
B *pb=new B;D *pd=dynamic_cast<D*>(pb);  
A *pa=new C;C *pc=dynamic_cast<C*>(pa);
```

某一系统功能,需要一次性加载 N (N 在 1000 左右) 个随机数,后续只对该集合进行遍历.最宜采用哪种结构存放?

正确答案: C 你的答案: 空 (错误)

Hash 表
二叉树
链表
图

将一颗多叉树存储在一个 txt 文件中,格式如下:

```
id1,parentId1,weight1  
id2,parentId2,weight2  
id3,parentId3,weight3  
.....
```

其中,一行表示一个节点,id 表示节点的序号,parentId 表示节点对应父节点的序号,weight 表示该节点的权重,

根节点的 parentId 是自身 id.请实现一个函数,输入是一个多叉树节点的数组和长度,要求打印出每一个节点的总权重

(总权重=节点自身权重+节点对应所有子节点的权重).自定义需要的数据结构,说明时间和空间复杂度(要求时间复杂度优先,空间复杂度尽量低)

```
1 #include <unordered_map>  
2 #include <iterator>  
3 #include <stdexcept>  
4 #include <set>  
5 #include <iostream>  
6 using namespace std;  
7  
8 typedef struct {  
9     size_t id;  
10    size_t parentId;  
11    size_t weight;  
12} Node;  
13  
14void print_weight_of_tree(Node array[], size_t len)  
15{  
16    if (len == 0) {  
17        return;  
18    }  
19}
```




```
19
20 unordered_map<size_t, size_t> id_index;
21 id_index.rehash(2 * len);
22 for(size_t i = 0; i != len; i++) {
23     id_index.emplace(array[i].id, i);
24 }
25
26 int *state = new int [len]();
27 size_t *weight = new size_t [len];
28 for(size_t i = 0; i != len; i++) {
29     weight[i] = array[i].weight;
30     //root node
31     if(array[i].id == array[i].parentId)
32         continue;
33     auto it = id_index.find(array[i].parentId);
34     if(it == id_index.end()) {
35         throw logic_error("parent not Found");
36     }
37     //tag
38     state[it->second] = 1;
39 }
40 bool end = false;
41 set<size_t> new_leaf_index;
42 while(!end) {
43     for(auto leaf_index : new_leaf_index) {
44         state[leaf_index] = 0;
45     }
46     new_leaf_index.clear();
47     for(size_t i = 0; i != len; i++) {
48         if(state[i] == 0) {
49             size_t parent = array[i].parentId;
50             //root node
51             if(parent == array[i].id)
52                 state[i] = -1;
53             end = true;
54             break;
55         } else {
56             size_t parent_index = id_index.find(parent)->second;
57             new_leaf_index.insert(parent_index);
58             weight[parent_index] += weight[i];
59             state[i] = -1;
60         }
61     }
62 }
```



```
63         }
64     }
65
66     for(size_t i = 0; i < len; i++) {
67         cout << "id: " << array[i].id << " weight: " << weight[i] <<
68endl;
69     }
70     delete [] state;
71     delete [] weight;
72 }
73
74 int main()
75 {
76     size_t len;
77     cin >> len;
78     Node *array = new Node[len];
79     for(size_t i = 0; i != len; i++) {
80         cin >> array[i].id >> array[i].parentId >> array[i].weight;
81     }
82     print_weight_of_tree(array, len);
83     delete [] array;
84     return 0;
85 }
```

思路：1、首先建立 `id_index` 的 `hash` 表，该表的作用为将节点 `id` 映射到节点在 `array` 中的下标，方便以后查询

2、算法的主要思想是先处理叶子节点（叶子节点 `state` 为 0），处理叶子节点时，将叶子节点的权重添加到其父节点上，并将叶子节点标记为删除（将对应的 `state` 设置为 -1），这时原来高度为 1 的所有节点又变成新的叶子节点，然后迭代处理，直到处理完根节点

3、其中 `state` 和 `weight` 分别为数组 `array` 中对应节点的状态(-1, 0, 1)和权重



更多
礼包
扫码关注



icebear.me

白熊事务所致力为准备求职的小伙伴提供优质的资料礼包和高效的求职工具。礼包包括**互联网、金融等行业的求职攻略**；**PPT模板**；**PS技巧**；**考研资料**等。

微信扫码关注：**白熊事务所**，获取更多资料礼包。

登陆官网：**www.icebear.me**，教你如何**一键搞定名企网申**。