算法专项练习：Python 程序练习 1

1.提取数字字符串中以逗号分隔的数字并转换为整数存入数组，再将数组中的元素进行分类，第一类为“小于 30”;第二类为“30～60”;第三类为“大于 60”。例如输入数字字符串为“34,23,45,99,24,56,9,87,”，输出结果为[23,9,24][34,45,56][87,99]。

(1)实现上述功能的 Python 程序段如下，请在划线处填入合适的代码。

```
s=input("请输入字符串(数字之间用逗号分隔):")
ch="";a=[]
for i in range(len(s)):
    if s[i]!=",":
        ch+=s[i]
    if s[i]="," or i==len(s)-1:
        a.append(int(ch))
        ① ch=""
i=0;j=len(a)-1;k=0
while k<=j:
    if ②a[k]<30:
        a[i],a[k]=a[k],a[i]
        i+=1
    elif a[k]>60:
        a[k],a[j]=a[j],a[k]
        j-=1
        ③ k+=1
    k+=1
print(a[:i],a[i:j+1],a[j+1:])
```

(2)若两组输入的数据分别为“12,78,65,7,45,2,55,”、“12,78,65,7,45,2,55”，则两组输出的结果__A__(单选，填字母:A.一致;B.不一致)。

2.在仓库有多个订单的货物，需要装车发货，订单信息保存在如下名为 dd 的列表中:

dd=[["订单 1","货物 F",6960],…,["订单 10","货物 C",7850]]

其中列表元素的含义为[订单名称，货物名称，货物重量]。

现在需要对这些订单的货物进行装车发货，遵循如下规则:

- 若单个订单的重量超过货车的最大载重量 maxzz，需要对其进行拆分，先装满整车，剩下的货物继续拆分装车；
- 拆分后余下的货物如果不能装满整车，则将较大的几个待装货物优先合并，装同一辆车；（不再继续打散）
- 有些货物之间有相互排斥关系，不能装同一辆车，如字典 hcdic{"货物 E":"货物 C","货物 C":"货物 E 货物 H","货物 H":"货物 C"}描述了货物之间的排斥关系。其中键值对"货物 C":"货物 E 货物 H”表示货物 C 不可以和货物 E 或货物 H 装在同一辆车；处理完后输出需要货车的数量、各订单货物装的车号、以及各车号所装的货物来源及重量。

程序运行样例如下:

maxzz=3000,订单及货物互斥信息如下:

[["订单 1","货物 B",800],["订单 2","货物 A",1500],["订单 3","货物 A",3500]]互斥数据:{"货物 C":"货物 D 货物 B","货物 D":"货物 C","货物 B":"货物 C"}一共需要 2 辆车。

各订单装货情况:订单 1 货物 B["车 2_800"]；订单 2 货物 A["车 2_1500"]；订单 3 货物 A["车 1_3000","车 2_500"]

各车辆装货情况:车 1_订单 3 货物 A3000；车 2_订单 2 货物 A1500,订单 1_货物 B800,订单 3_货物 A500

(1)若订单及货物互斥信息如下，货车最大载重量为 3000，需要准备__3__辆货车。

[["订单 1","货物 B", 800],["订单 2","货物 A",1500],["订单 3","货物 A",3500]]

互斥数据:{"货物 A":"货物 D 货物 B","货物 D":"货物 A","货物 B":"货物 A"}A B 互斥

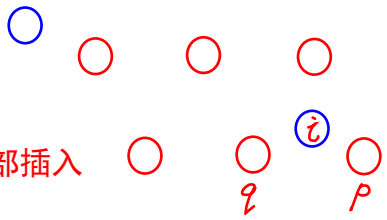
车1
订单3 货物A 3000
车2
订单2 货物A 1500
订单3 货物A 500
车3
订单1 货物B 800

(2)实现题目描述的 Python 程序如下，请完成填空。

```
maxzz=3000;che=[];n=10#maxzz 是车的最大载重量；che 是各货车装载货物详情,n 是订单数量
```

```
#设定订单的信息 dd,格式如下:
#dd=[["订单 1","货物 A",800],……,["订单 10","货物 C",3500]]
#设定互斥(不能同车发送)货物信息 hcdic,格式如下:
#hcdic={"货物 A":"货物 B","货物 B":"货物 A 货物 C","货物 C":"货物 B"},代码略
def sort_link(link):  #对链表 link 降序排列
    head=0
    link[head][-1]=-1
    for i in range(1,n):
        q,p=-1,head
        while ①p!=-1 and link[i][2]<link[p][2]:
            q,p=p,link[p][-1]
        if q==-1:
            link[i][-1],head=p,i
        else:
            link[i][-1],link[q][-1]=p,i
    return head
cheid=0
for i in range(n):    #大件货订单先处理
    dd[i]+=[[],0]
    while dd[i][2]>=maxzz:
        cheid+=1
        #记录本订单拆分情况
        dd[i][3].append("车"+str(cheid)+"_"+str(maxzz))
        #记录本车货物来源
        che.append("车"+str(cheid)+"_"+dd[i][0]+"_"+dd[i][1]+"_"+str(maxzz))
        ②dd[i][2]-=maxzz
head=sort_link(dd)
while ③dd[head][2]!=0 或 dd[head][2]>0:
    total=0
    hcstr=""
    cheid+=1
    txt="车"+str(cheid)+"_"
    p=head
    while p!=-1 and dd[p][2]>0:
        if ④dd[p][2]+total<=maxzz and dd[p][1] not in hcstr:  #合并货物
            dd[p][3].append("车"+str(cheid)+"_"+str (dd[p][2]))
            txt+=dd[p][0]+"_"+dd[p][1]+"_"+str(dd[p][2])+","
            if dd[p][1] in hcdic:
                hcstr+=hcdic[dd[p][1]]
            total+=dd[p][2]
            dd[p][2]=0
        p=dd[p][-1]
    che.append(txt)
    head=sort_link(dd)
print("一共需要"+str(cheid)+"辆车。")
print("各订单装货情况:")
for i in range(len(dd)):
    print(dd[i][0],dd[i][1],dd[i][3])
print("各车辆装货情况:")
for item in che:
    print(item)
```

链表降序排序：
将列表种元素逐个插入到降序列表中



dd[i]添加 [] 和指针域

只考虑订单中超重的情况

for循环后
物品不再拆分

推测：即使重量刚好为maxzz，剩余重量为0，dd链表中也不会删除节点

错误写法：head!=-1
链表长度不会变化，链表各节点剩余重量最终会减少到0，每次while循环结尾，重新对dd链表降序排序

hcdic[dd[head][1]]

将不能运输的货物信息合并加入到hcstr
本次车可能运送有很多物品，各自对应的互斥物品都加到hcstr （用列表数据结构可能更佳）