高三下信息算法综合练习1

1. 某机构采购了 n 类（编号 1～n）紧急救援食品，所有食品均采用统一的包装袋进行封装，食品储存条件分冷藏和常温。现将食品存放在若干货柜内，每个货柜的最大容量为 m 袋，货柜编号从 1 开始。存放时，首先按类别编号依次在各类食品内处理，将每 m 袋存放于一个新的货柜，装袋数量不足 m 的暂不处理；然后按剩余食品袋数由多到少的顺序，依次为各类食品按照储存条件分配存放的货柜。

若某类食品的剩余袋数为 k，则存放过程为：在已存放食品的货柜内寻找空余空间不少于 k 的货柜，且满足已存食品与当前食品的储存条件相同（冷藏或常温）。若存在多个这样的货柜，则挑选空余空间最小的来存放，若不存在，则存放于新的货柜。

设n 为4，m 为100，当食品袋数超过m 时，货柜的安排过程如第1 图a 所示，各类剩余食品的放置过程如第1 图b 所示。其中编号为2 的食品需冷藏。

类别编号	装袋数量
1	110
2	90
3	165
4	275

类别编号	剩余袋数	货柜编号
1	10	1
2	90	
3	65	2
4	75	3,4

食品装袋数量超过m时

第 1题图 a

类别编号	剩余袋数	货柜编号
2	90	
4	75	3,4
3	65	2
1	10	1

类别编号	剩余袋数	货柜编号
2	90	5
4	75	3,4,6
3	65	2,7
1	10	1,6

对剩余食品的放置

第 1题图 b

编写程序：给定各食品类别编号及装袋数量，根据上述方法进行放置，按食品类别编号次序输出各类食品所放置的货柜编号。请回答下列问题：

（1）由题意可知，若仅将第14 题图 a 中类别 3 的装袋数量改为 185，然后对图中 4 类食品重新放置，则类别 3 所放置的货柜编号为__▲__ 。

（2）定义 getroom(a,key,f)函数，参数 a 中每个元素由货柜的空位数和储存条件（0 常温，1冷藏）两个数据项构成。函数的功能是为各类剩余食品寻找储存的货柜编号。

```
def getroom(a,key,f):
    flag = True
    i=0
    while i < len(a):
        if a[i][0] >= key and a[i][1]==f:
            if flag or a[i] <= a[m]:
                m=i
                flag = False
        i+=1
    if flag==False:
        return m
    else:
        return len(a)
```

若 a 为[[0,0],[25,1],[30,0],[20,1],[15,0]],key 为 15，f 为 1，调用 getroom(a,key,f)后，返回的值为__▲__。

（3）实现上述功能的部分 Python 程序如下，请在程序中划线处填入合适的代码。

```
def group(data, m):
    n=len(data)
    rooms=[]   #rooms[i]存放 i 类食品放置的货柜编号
    b=[[0,0]]
    for i in range(n+1):
        rooms. append([])   #在列表 rooms 末尾追加一个元素[]
    rnum=0
    for i in range(n):   #处理装袋数量超过 m 的食品
        while data[i][1]>=m:
            rnum+=1
            b. append([0,data[i][2]])
                    ①
            rooms[k]. append(rnum)
            data[i][1]-=m
    #根据各类食品的剩余袋数，对 data 进行降序排序，代码略i=0
    while   i<n :
        rest=data[i][1]
        marker=data[i][2]
        dyroom=getroom(b,rest,marker)
        if          ②          :
            rnum+=1
            b. append([m-rest,data[i][2]])
            rooms[data[i][0]]. append(rnum) else:
                    ③
            rooms[data[i][0]]. append(dyroom)
        data[i][1]=0
        i+=1
    #输出各类食品放置的货柜编号，代码略
'''
读取货柜的最大容量存入m；读取编号为1～n 食品数据，依次存入列表data 的data[0]至data[n-1]。
data[i]包含 3 个数据项，data[i][0],data[i][1],data[i][2]分别存放类别编号、装袋数量以及储存条件（0 常温，1 冷藏），如示例：data=[[1,110,0],[2,90,1], . . .],即编号为 2 的食品装袋数量为 90，需放置冷藏货柜。
'''
group(data, m)
```

2.计算机运行多个任务（又称进程）时，需要进行调度。有的进程需要优先响应，例如用户的交互操作，此时就需要暂停当前运行的进程，让CPU先执行需要优先响应的进程，这称为抢占。操作系统需要设计调度算法，来决定CPU运行进程的顺序。

优先级抢占式调度算法是一种简单的调度算法，规则如下：

1）将进程分为m个优先级，设置m个等待队列，分别对应每一级优先级。

2）每个进程具有三个要素：到达时间，运行所需时长，优先级数（数越大优先级越高）

3）相同优先级的进程，按照先到先服务的原则依次执行。

4）同一时刻中，先将到达的进程都加入队列，再按照优先级进行分配

5）只有当k级队列为空的时候，才会为k-1级队列队首的进程分配时间。

6）进程Pi运行时，如果有优先级更高的进程Pj到达，则立即发生抢占，先执行Pj,并将进程Pi剩余未执行完的部分，重新加入Pi优先级对应的队列末尾，等待继续执行。

编写程序模拟CPU分配计算资源的过程，已知按照到达时间升序排序的进程数据(包含到达时间、运行时长、优先级)，计算并输出每个进程最终处理完成的时间。（时间单位均为毫秒）。

请回答下列问题：

（1）有4个进程A、B、C、D如题表1所示。

进程	到达时间	运行时长	优先级
A	0	7	1
B	2	4	2
C	4	1	3
D	5	4	2

表1

由优先级抢占式调度算法的规则可知，0毫秒时进程A到达并执行；2毫秒时进程B到达，B的优先级高于A，发生抢占，A剩余的5毫秒回到队列1,B开始执行；4毫秒时进程C 到达，C的优先级高于B，发生抢占，B剩余的2毫秒回到队列2，C开始执行；则进程D执行完的时刻为

____▲_____。

（2）模拟实现优先级抢占式调度算法Python程序如下，请在划线处填入合适的代码。

```
def insert(p,remain):
    data[p][2]=remain        #更新进程剩余的运行时间
    lvl=data[p][3]
    if queinfo[lvl][0]==-1:
        queinfo[lvl][0]=p
    if queinfo[lvl][1]!=-1:
        data[queinfo[lvl][1]][-1]=p
    data[p][-1]=-1
    queinfo[lvl][1]=p
m=int(input("设置优先级的数量m:"))
# 输入列表data 存储进程，data中的节点包含信息有[名称，到达时间，运行时长，优先级]，代码略
# 进程已经按到达时间升序排序
# 例如：data=['A',0,7,1],['B',2,4,2],['C',4,1,3],['D',5,4,2]]
```

```
for i in range(len(data)):
    data[i].append(-1)
queinfo=[[-1,-1] for i in range(m+1)]
insert(0,data[0][2])        #将第1个进程加入队列
time=data[0][1]
cnt=1                       #所有队列内等待的进程总数
idx=1
lvl=m
while cnt> 0:
    if queinfo[lvl][0]!=-1:
        cur=queinfo[lvl][0]
        queinfo[lvl][0]=data[queinfo[lvl][0]][-1]
        cnt-=1
        ①_____________________________
        while idx<len(data) and time+data[cur][2]>=data[idx][1]:
            if lvl>=data[idx][3] or time+data[cur][2]==data[idx][1]:
                insert(idx,data[idx][2])
                cnt+=1
                idx+=1
            elif time+data[cur][2]>data[idx][1]:
                insert(idx,data[idx][2])                #抢占的进程也先入队
                cnt+=1
                insert(cur,②_____________________________))
                cnt+=1
                time=data[idx][1]
                lvl=data[idx][3]
                idx+=1
                flag=True
                break
        if flag==False:
            time=time+data[cur][2]
            print("时刻", time, ":进程", data[cur][0], "完成")
            lvl= m
            if  ③_____________________________: #仍然有未到达的进程等待入队
                insert(idx,data[idx][2])
                cnt+=1
                time=data[idx][1]
                idx+=1
    else:
        lvl-=1
        if lvl==0:
            lvl=m
```

（3）若将以上程序中insert函数内的加框处代码删除，会导致某些情况下无法得到符合程序功能的结果，下列4组数据中能测试出这一问题的是__▲__（单选，填字母）

A. m = 3 data=[['A', 0, 1, 1], ['B', 1, 1, 2], ['C', 3, 1, 3]]	B. m = 3 data= [['A', 0, 2, 2], ['B', 1, 2, 3], ['C', 3, 1, 3]]
C. m = 3 data=[['A', 0, 2, 1], ['B', 1, 2, 3], ['C', 2, 1, 2]]	D. m = 3 data = [['A', 0, 2, 2], ['B', 1, 2, 3], ['C', 2, 1, 1]]