高三下信息选考算法综合——链表

【25.5 学军考前模拟】.为加强基础学科拔尖创新人才选拔培养，某校采用综合招生模式录取，在高考之外，额外进行校测，其中校测包括学科基础素质测试和综合素质测试。考生综合成绩=高考成绩(按满分 85 分折算)+学科基础素质测试成绩(按满分 10 分折算)+综合素质测试成绩(按满分 5 分折算)。

考生在报名时可以选择两种类型，若作为第 1 类考生报名，根据各省份的招生计划数的 6 倍按高考成绩从高到低(末位同分同入围)确定入围综合招生的考生名单，入围后参加各项测试，根据各省份招生计划数，按综合成绩由高到低择优确定录取名单，若考生综合成绩相同，则按以下单项顺序及分数高低排序进行录取：高考成绩、校测总分、学科基础素质测试成绩、综合素质测试成绩。若作为第 2 类考生报名，提交特长申请后经审核入围，2 类考生有单独的全国统一招生计划数，按照考生综合成绩相对值在全国范围内降序排序录取，综合成绩相对值=考生综合成绩-所在省份 1 类考生最低录取综合成绩，若相对值相同，则按以下单项顺序及分数高低进行排序：校测总分、学科基础素质测试成绩、综合素质测试成绩。为简化问题，假设不存在两个成绩一模一样的考生。请回答下列问题：

(1)若 A 省和 B 省的 1 类生招生计划各 1 人，2 类生的全国招生计划为 1 人，有以下 5 位同学入围综合招生，则录取的同学为11、18、25(填考生报名号)。

报名号	省份	考生类型	综合成绩	高考成绩
9	A	1 类生	97	79
11	A	1 类生	97	81
16	A	2 类生	96	78
18	B	1 类生	93	79.5
25	B	2 类生	93	75

(2)列表 rec 存储某省的 1 类生报名记录，每个元素包含报名号、高考成绩 2 个数据项，有如下 Python 程序：

```
def select(rec,cnt): # 根据某省的报名记录返回成功入围的考生名单
    n = len(rec)
    lm = cnt *6 # 按 6 倍招生计划确定入围基准
    i=0
    while i <n:
        for j in range(n-1,i,-1):
            if rec[j][1] > rec[j-1][1]:
                rec[j], rec[j-1] = rec[j-1], rec[j]
        if i >= lm and rec[i][1] < rec[i-1][1]:
            break
        i +=1
    return rec[:i]
```

① 若调用上述 select 函数，其中参数 rec = [[1,693],[2,691],[3,688],[4,700],[5,683],[6,688],[7,694],[8,682],[9,696],[10,680]]，参数 cnt=1，则加框处代码执行次数为 8

② 将下划线处代码改为下列选项中的一项 D (单选，填字母)，对程序行为没有影响。

A. rec[i][1] <= rec[i-1][1]　　B. rec[i][1] > rec[i-1][1]

C. rec[i][1] >= rec[lm -1][1]　　D. rec[i][1] != rec[lm -1][1]

(3)下面的 Python 程序模拟了考生录取过程，最终输出录取的考生信息，请在划线处填入合适的代码。

```
info = {"浙江":[-1,-1,80],"江苏":[-1,-1,60],…} #80 为浙江省 1 类生的招生计划数，依次类推
plan =200     #2 类生招生计划数
shortlist = []
for prv in info:
    #读取该省 1 类生报名数据至 rec 列表中，代码略
    pass_rec = select(rec, info[prv][2])
    shortlist += pass_rec
    #读取该省 2 类生审核后入围考生信息存入 shortlist 列表中，代码略
#读取 shortlist 中的每一位考生测试成绩存入列表 data 中，data 每个元素包括 9 个数据项，依次为：报名号、姓名、省份、考生类型(0 表示 1 类生，1 表示 2 类生)、综合成绩、高考成绩、校测总分、学科基础素质测试成绩、综合素质测试成绩，代码略
```

```
def compare(x, y):
    for i in range(5 +x[3],9):
        if x[i]!=y[i]:
            return x[i]>y[i]
def rel_value(prv):
    return data[info[prv][1]][4]- data[info[prv][0]][4]
def greater(p, q):
    if q=="":
        return True
    if p=="":
        return False
    vp = rel_value(p)
    vq= rel_value(q)
    if vp != vq:
        return vp>vq
    else:
        return compare(data[info[p][1]], data[info[q][1]])
n = len(data)
for i in range(n):
    data[i].append(-1)
    prv, tp = data[i][2], data[i][3]
    if info[prv][tp]==-1:
        info[prv][tp]=i
    else:
        pre = info[prv][tp]
        p=pre
        while p!= -1:
            if data[p][4]<data[i][4]:
                break
            elif data[p][4]== data[i][4] and __①____________:
                break
            pre=p
            p = data[p][-1]
        if p==pre:
            info[prv][tp]=i
        else:
            data[i][-1]=p
        data[pre][-1]=i
for pry in info:
    p= info[prv][0]
    while p != -1 and info[prv][2]> 0:
        print(data[p][0], data[p][1],"恭喜录取。")
        info[prv][2]-=1
        __②____________
        p = data[p][-1]
while plan >0:
    p1=p2=""
    for prv in info:
        if info[prv][1]!= -1:
            if greater(prv, p1):
                p2 =p1
                p1= prv
            elif greater(prv, p2):
                p2 =prv
    if p1=="":
        break
    while _③____________:
        print(data[info[p1][1]][0], data[info[p1][1]][1],"恭喜录取。")
        info[p1][1]= data[info[p1][1]][-1]
        plan -=1
```

compare(data[i],data[p])

info[prv][0] = p

记录prv省最后一名，为2类考生算综合相对值准备

找出2类考生分数最高的两个省（也可能只有一个省）

info[p1][1]!=-1 and plan>0 and greater(p1,p2)

如果p1省的前几名都比p2省高，就一直取