

高三上信息选考练习卷10（台州一模）

一、选择题（本大题共 12 题，每题 2 分，共 24 分。每题列出的四个备选项中只有一个是符合题目要求的，不选、多选、错选均不得分）

阅读下列材料，回答第 1 至 3 题。

某博物馆引入智能导览系统，该系统通过游客手持的导览器获取游客的实时位置，推送当前展品的图文介绍、语音解说及视频资料。游客可预先选择感兴趣的展品类别，系统将其规划个性化的游览路线。

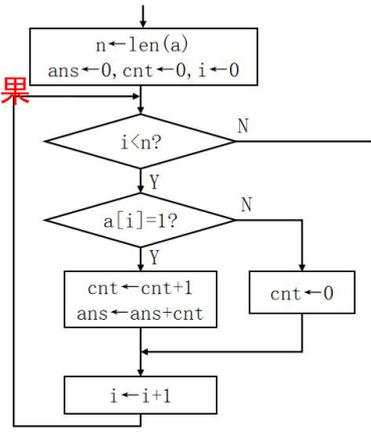
- 下列关于该系统中数据与信息的说法，正确的是 **C**
 - 同一展品对不同游客的价值相同
 - 系统推送展品资料不需要传输介质
 - 系统中的数据是以二进制形式存储的
 - 游客在游览过程中不会产生新的数据
- 下列有关信息安全与保护的做法，合理的是 **A**
 - 定期备份博物馆展品数据
 - 闭馆期间关闭系统服务器防火墙
 - 将游客的相关信息分享给其他博物馆
 - 在展厅大屏幕上显示所有游客的实时位置
- 下列关于该系统中数据处理的说法，不正确的是 **D**
 - 分析游客的游览路线可为优化展品布局提供依据
 - 导览器播放展品的语音解说，需要经过数模转换
 - 新展品的图文介绍等资料需经数字化后才能被系统推送
 - 为减少系统存储占用，将 MP3 格式的音频转换成 WAV 格式 **WAV->MP3是压缩**

阅读下列材料，回答第 4 至 6 题。

某校部署智慧体测系统，该系统配备了显示屏、摄像头等设备。学生站立于指定点位后，触发系统的人脸识别功能，实现身份认证。随后，学生用手势选择体测项目进行练习，系统通过摄像头实时捕捉学生的动作姿态进行评分，并将生成的运动数据上传至服务器。练习结束后，系统自动处理数据并生成分析报告。家长可通过手机端 APP 查看分析报告。

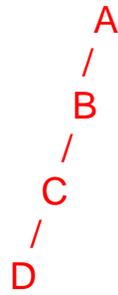
- 下列关于该系统功能与软件设计的描述，正确的是 **A**
 - 设计系统时需要考虑数字鸿沟问题
 - 系统生成的分析报告存储在手机端 APP 中
 - 通过显示屏呈现运动数据属于数据输入功能
 - 通过人脸识别实现身份认证无需软件的支持
 - 下列关于该系统硬件和网络的说法，正确的是 **B**
 - 该系统的硬件只有显示屏和摄像头
 - 服务器的存储容量会影响系统性能
 - 上传数据至服务器无需遵循网络协议
 - 通过移动通信网络才能在手机端 APP 查看分析报告
 - 系统的人脸识别功能是基于卷积神经网络方法实现的，学生仅在佩戴口罩时易导致识别失败，下列原因最有可能的是 **D**
 - 识别人脸的知识规则和规则不够完善
 - 在模型训练完成后未保留原始训练数据 **要保留训练结果**
 - 卷积神经网络模型的训练参数设置不合理
 - 口罩遮挡导致提取的人脸特征数据不完整
7. 某算法的部分流程图如第 7 题图所示，若列表 a 的值为 [1, 0, 1, 1, 0, 0, 1, 1, 1, 0]，执行这部分流程后，ans 的值为 **C**
- 3
 - 6
 - 10
 - 21

cnt统计连续的1的个数
ans=1+1+2+1+2+3=10

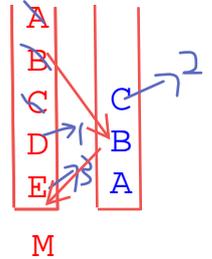


第 7 题图

8. 某二叉树的前序遍历结果为 ABCD, 下列说法正确的是
- A. 该二叉树的深度一定为 3 **可以是3, 也可以是4**
 - B. 节点C一定是节点D的父节点
 - C. 该二叉树共有 6 种不同的形态
 - D. 该二叉树的后序遍历结果可能为 DCBA



9. 现有 M 和 N 两个栈, 栈 M 从栈顶到栈底的元素依次为 ABCDE, 栈 N 初始为空。每次操作可将其中一个栈的栈顶元素出栈后输出或放入另一个栈。若要输出序列为 DCEAB, 则元素 B 需要经历的操作次数至少是 **B**
- A. 2
 - B. 3
 - C. 4
 - D. 5



10. 有如下 Python 程序段:
- ```
for i in range(len(data)):
 p = i; cnt = 0
 while p != -1:
 cnt = cnt + 1
 p = data[p][1]
 if cnt == len(data):
 head = i
```

**B从M到N,再回M, 出栈**

**找链头, 从当前节点遍历到链尾  
若当前节点遍历到链尾的个数是总长度, 就找到链头**

若 data 为  $[[6, 1], [8, -1], [1, 3], [5, 0], [9, 5], [7, 2]]$ , 运行该程序段后, head 的值为 **B**

- A. 3
- B. 4
- C. 5
- D. 7

11. 有如下 Python 程序段:

```
L = 0; R = len(a) - 1
while L < R:
 m = (L + R) // 2
 if a[m] < a[m + 1]:
 L = m + 1
 else:
 R = m
print(L)
```

**快速找链头方法:  
没有指针指向头结点  
非头结点都有指向它的指针  
所以:  
看指针域中数字没有出现的, 一般是链头**

**对分变形  
找序列中的最大值所在位置**

运行该程序段后, 输出结果为 4, 则列表 a 不可能是 **D**

- A.  $[1, 3, 5, 6, 8, 4, 2]$
- B.  $[3, 5, 6, 8, 9]$
- C.  $[1, 8, 7, 4, 8, 6, 5]$
- D.  $[1, 3, 5, 8, 8, 7, 4]$

12. 列表 s 包含 n 个元素, 现要对列表 s 进行消除操作: 选择两个相邻且相同的元素进行删除。反复执行该操作, 直到无法继续消除。例如: 列表 s 为  $["a", "b", "b", "a", "c", "a"]$ , 完成所有消除操作后, 结果为  $["c", "a"]$ 。实现上述功能的 Python 程序段如下:

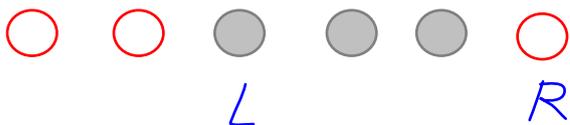
```
left, right = 0, 0
while right < len(s):
 s[left] = s[right]
 if left > 0 and s[left] == s[left-1]:
 (1) left = left - 1
 else:
 (2) left = left + 1
 (3) right = right + 1
print((4) s[0:left])
```

上述程序段中方框处可选代码为: **C**

- ① left = left + 1
- ② left = left - 1
- ③ right = right + 1
- ④ right = right - 1
- ⑤ s[0:left]
- ⑥ s[0:left+1]

- 则 (1) (2) (3) (4) 处代码依次为
- A. ①②③⑤
  - B. ①②④⑥
  - C. ②①③⑤
  - D. ②①③⑥

推测 left-right-1 区段为已经删除的数据



二、非选择题（本大题共 3 题，其中第 13 题 7 分，第 14 题 10 分，第 15 题 9 分，共 26 分）

13. 有一条从西向东延伸、长度为  $m$  公里的直线型公路，沿线有  $n$  个适合建设基站的点位，点位的位置以其距离公路最西端的距离表示。每个基站的信号覆盖半径为  $r$  公里，即选择位置为  $x$  的点位建设基站，信号可覆盖  $x-r$  到  $x+r$  范围的公路。现要选择部分点位建设基站，使整段公路（从 0 到  $m$ ）被信号覆盖。编写程序，根据点位位置，输出最少需要建设的基站数量；若无法实现整段公路的信号覆盖，则输出 -1。请回答下列问题：

(1) 若公路长度为 10 公里，每个基站的信号覆盖半径为 3 公里，有 3 个适合建设基站的点位，位置分别为 2、7、9，则最少需要安装的基站数量为 2。

(2) 实现上述功能的部分 Python 程序如下，请在划线处填入合适的代码。 **2和7**

```

#输入公路长度 m、基站信号覆盖半径 r、点位数量 n，代码略
#读取 n 个点位的位置，升序排序后存入列表 a，代码略
ans, i = 0, 0
pos = 0
while pos < m:
 d = -1
 while i < n and a[i] - r <= pos:
 d = a[i]
 i = i + 1
 if d == -1:
 break
 pos = d + r
 ans = ans + 1
if d != -1 or pos >= m:
 print(ans)
else:
 print("-1")

```

算法1：从  $pos+d$  开始分析

算法2：从 0 开始分析

本题哪种算法？

找到第一个能覆盖到  $pos$  位的站点，找到之后  $pos$  前移  $r$  距离，继续找能覆盖新的  $pos$  位的站点，直到  $pos \geq m$ ，表示所有区域都能覆盖到

站点只能建立在某个  $a[i]$  上，不能建在  $pos$ 、 $pos+d$

if  $d \neq -1$  或  $pos \geq m$

14. 某城市搭建了道路积水监测系统，在各主要道路设置监测点。各监测点的传感器每隔 5 分钟采集 1 次积水深度数据，智能终端从服务器获取阈值，根据该阈值和采集的数据控制执行器发出预警信号，并通过网络将采集的数据传输至服务器，存储到数据库中。用户通过浏览器可查看系统数据。请回答下列问题：

(1) 在搭建该系统时，无需在各监测点部署的设备是 C（单选，填字母）。

- A. 传感器                      B. 智能终端                      C. 服务器                      D. 执行器

(2) 下列关于该系统的说法，正确的是 AB（多选，填字母）。（注：全部选对的得 2 分，选对但不全的得 1 分，不选或有错的得 0 分）

- A. 智能终端与服务器的数据传输是双向的  
 B. 可通过设置传感器编号区分数据的监测点来源  
 C. 若要调整采集数据的时间间隔，应修改服务器端程序 **在智能终端上改**  
 D. 用户通过浏览器查看系统数据需要知道数据库的文件名 **不需要**

(3) 为提升该系统的稳定性和可靠性，下列措施合理的是 ACD（多选，填字母）。

（注：全部选对的得 2 分，选对但不全的得 1 分，不选或有错的得 0 分）

- A. 为智能终端配备备用电源  
 B. 对所有监测点设置相同的积水深度预警阈值  
 C. 选用高精度、具备防水和抗干扰能力的传感器  
 D. 若数据传输失败，将数据暂存智能终端并尝试再次发送

不建议：  
降低摄像头采集频率  
优化服务器算法  
提升网络带宽  
更换更好的服务器  
降低摄像头采集分辨率

## 用更好的压缩算法对图像进行压缩 定期清理服务器中的数据

(4) 为了更直观地了解各监测点出现预警信号时的情况，在各监测点增配摄像头，每隔 5 分钟采集一次图像数据，通过智能终端发送给服务器。运行一段时间后，发现图像数据传输占用了大量网络带宽，服务器存储空间变得紧张，且响应速度迟缓。请写出一种可行的改进方法。

采集的图像数据进行压缩处理，并分批上传至服务器。  
或 触发预警信号时，智能终端控制摄像头采集图像数据并传输至服务器。

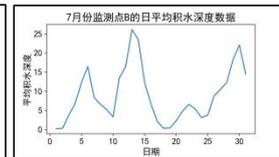
(5) 将系统中某年的数据导出到文件 data.xlsx 中，部分数据如第 14 题图 a 所示。其中“是否预警”列是指执行器是否发出预警信号，1 表示发出，0 表示未发出。现要由高到低输出 7 月份各监测点执行器发出预警信号的次数（如图 b 所示），再用预警次数最高的监测点的 7 月份日平均积水深度数据绘制线形图（如图 c 所示）。

| 监测点 | 年    | 月 | 日  | 时间点   | 积水深度 | 是否预警 |
|-----|------|---|----|-------|------|------|
| A   | 2024 | 6 | 12 | 07:10 | 15.3 | 1    |
| B   | 2024 | 6 | 12 | 07:10 | 12.7 | 0    |
| C   | 2024 | 6 | 12 | 07:10 | 4.5  | 0    |
| D   | 2024 | 6 | 12 | 07:10 | 4.5  | 0    |
| E   | 2024 | 6 | 12 | 07:10 | 18.2 | 1    |

第 14 题图 a

|                   |
|-------------------|
| 监测点: B 预警次数: 1626 |
| 监测点: A 预警次数: 909  |
| 监测点: C 预警次数: 632  |
| 监测点: D 预警次数: 499  |
| .....             |

第 14 题图 b



第 14 题图 c

实现上述功能的 Python 程序如下，请选择合适的代码填入划线处（填字母）。

#导入相关库、设置字体，代码略

df = pd.read\_excel("data.xlsx")

df1 = df[df["月"] == 7]

统计预警次数不能用 count()

df2 = E. df1.groupby("监测点", as\_index = False).是否预警.sum()

df2 = D. df2.sort\_values("是否预警", ascending = False) 按预警次数降序排序

#依次输出 df2 中各监测点编号及预警次数，如图 b 所示，代码略

#将 df2 首行的监测点编号存入 uid，代码略

df2 = F. df1[df1.监测点 == uid]

df2 = df2.groupby("日", as\_index = False).积水深度.mean()

plt.plot(df2.日, df2.积水深度)

#设置绘图参数，并显示如图 c 所示的线形图，代码略

①②③处可选的代码有：

A. df2.sort\_values("预警次数", ascending = False)

B. df1.groupby("监测点", as\_index = False).是否预警.count()

C. df[df.监测点 == uid]

D. df2.sort\_values("是否预警", ascending = False)

E. df1.groupby("监测点", as\_index = False).是否预警.sum()

F. df1[df1.监测点 == uid]

就是为了找uid

15. 某工厂接到一批零件订单，要用一台设备进行加工。每个订单的信息包含订单编号、加工时长、截止时间和订单状态。订单状态为“T”表示设备符合该订单的加工要求；标记“F”表示设备不符合该订单的加工要求。

该工厂从第1天开始工作，在订单状态为“T”的订单中选择尽可能多的订单依次进行加工，一旦选择某个订单，其加工结束时间必须小于等于该订单的截止时间，且处理完该订单后才能开始处理下一个订单。（时间单位均为天）

从按截止时间升序排序看：  
可能是贪心算法  
→截止时间越早优先安排

| 订单编号 | 加工时长 | 截止时间 | 订单状态 |
|------|------|------|------|
| S01  | 5    | 7    | T    |
| S02  | 3    | 10   | F    |
| S03  | 8    | 13   | T    |
| S04  | 4    | 14   | F    |
| S05  | 7    | 18   | T    |

S01 1~5  
S03 安排  
S04 无法按时完成，换掉S03 (s03耗时>s04)  
S05 安排

第 15 题图 a

例如，工厂接到一批零件订单信息如第 15 题图 a 所示，最多可完成的订单数量为 2 个，可能的加工方案如第 15 题图 b 所示。

| 时间(天) | 1   | 2 | 3 | 4 | 5 | 6   | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|-------|-----|---|---|---|---|-----|---|---|---|----|----|----|----|
| 加工订单  | S01 |   |   |   |   | S03 |   |   |   |    |    |    |    |

第 15 题图 b

编写程序，根据给定的订单信息，计算该工厂最多可完成的订单数量，并按截止时间升序输出这些订单的编号。若有多种可行方案，则优先选择最早完成所有订单的方案。请回答下列问题：

- 由题意可知，若将第 15 题图 a 中编号 S04 的订单状态修改为“T”，则该工厂最多可完成的订单数量为 3。
- 定义如下 bubble\_sort(lst) 函数，参数 lst 列表中每个元素包含 4 个数据项，依次为订单编号、加工时长、截止时间和订单状态。函数的功能是根据每个订单的截止时间，对 lst 进行升序排序，并删除状态为“F”的订单。

```
def bubble_sort(lst):
 i = 0; n = len(lst)
 while i < n:
 for j in range(n - 1, i, -1):
 if lst[j][3] == "T":
 if lst[j - 1][3] == "F" or lst[j][2] < lst[j - 1][2]:
 lst[j], lst[j - 1] = lst[j - 1], lst[j]
 if lst[i][3] == "F":
 break
 i = i + 1
 return lst[0:i]
```

类似男生按成绩升序排序  
女生在前，男生乱序在后  
代码优化

T T F T F  
第一趟 i=0: s01, s03, s02, s05, s04  
T T T F F  
第二趟 i=1: s01, s03, s05, s02, s04  
第三趟, i=2, lst[i][3] != 'F', i=i+1 再执行一次  
第四趟, i=3, lst[i][3] == 'F', i=i+1 不执行

- 若 lst 列表依次存储第 15 题图 a 所示的订单信息，如 lst[0] = ["S01", 5, 7, "T"]，调用 bubble\_sort(lst) 函数，则语句“i = i + 1”的执行次数为 3。
- 若将加框处语句修改为“i < n - 1”，则对函数返回结果 A (单选，填字母：A. 有影响 / B. 无影响)

如果所有记录为T，则return s[0:i]就少了一条

(3) 实现上述功能的部分 Python 程序如下，程序中用到的列表函数与方法如第 15 题图 c 所示，请在划线处填入合适的代码。

| 函数与方法       | 功能                         |
|-------------|----------------------------|
| w.append(x) | 在列表 w 末尾添加元素 x             |
| x=w.pop()   | 将列表 w 末尾元素赋值给 x，并将其从 w 中删除 |
| w.remove(x) | 将列表中值为 x 的元素删除             |

第 15 题图 c

```

#读取订单加工时长上限存入 m, 代码略
#读取待处理的订单信息存入 data 列表, 代码略
data = bubble_sort(data)
a = [] #存放工厂选择处理的订单编号
c = []
for i in range(m + 1): m对应订单加工时长
 c.append([]) c的作用类似“桶”
current = 1
ans = 0
for i in range(len(data)):
 if ① current+data[i][1]-1<=data[i][2] 可以在截止前完成的
 任务
 c桶, 存储订单编号 c[data[i][1]].append(data[i][0]) c[任务时长]列表中加入订单编号
 a存储订单编号 a.append(data[i][0]) (同一时长的按编号先后加入)
 current += data[i][1] 时间调整到该任务结束
 ans += 1
 else: 如果无法在截止前完成这个任务, 考虑去掉耗时最长的来替换
 j = m j从最大加工时长开始
 while j >= 0 and len(c[j]) == 0: 找目前已经在c中出现的耗时最长的任务
 j-=1 ②
 if j >= 0 and data[i][1] < j: 找到耗时最长的任务, 如果当前任务的
 v: 订单编号 v = c[j].pop() 时长小于原耗时最长的任务, 则删除原
 a.remove(v) 耗时最长的任务, 用当前任务来替换
 c[data[i][1]].append(data[i][0])
 a.append(data[i][0])
 current = current-j+data[i][1] ③ 更新时间, 将原耗时j天的任
 务去掉, 加上当前i所在任务
 的时长data[i][1]
#输出最多可处理的订单数量和对应的订单编号, 代码略

```

已排任务加工时长 (桶)

1  
2  
3  
4  
5 s01  
6  
7  
8 s01

v: 订单编号

按截止时间升序排序,

s03→遍历s04时

s04截止时间 (14) > s03截止时间 (13) ,

加工时长 4 < s03 (8)

s04换掉s03