绝密★考试结束前

Z20⁺名校联盟（浙江省名校新高考研究联盟）2026 届高三第一次联考

技术试题卷

信息命题：慈溪中学 张佳晨、龚益科
磨稿：玉环中学 杨倩倩 海宁高级中学 张 星 黄岩中学 奚敏丽 校稿：刘洪杰、徐 江
通用命题：黄岩中学 梁 峰、郑炳雯
磨稿：慈溪中学 马功平 嵊州中学 姚维红 龙湾中学 张政儒 校稿：赵 刚、戴月凤

考生须知：

1. 本卷满分 100 分，考试时间 90 分钟。
2. 答题前，在试卷指定区域填写学校、班级、姓名、试场号、座位号及准考证号。
3. 所有答案必须写在答题卷上，写在试卷上无效；考试结束后，只需上交答题卷。

第一部分：信息技术（共 50 分）

一、选择题（本大题共 12 小题，每小题 2 分，共 24 分。每小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 4 题。

某市建设智慧健身步道系统，在步道沿线安装灯杆。该系统集成环境监测、AI 人流量统计、语音紧急呼叫等功能。市民可通过手机 APP 查看实时空气质量文字提示、步道人流量热力图，并参与线上健身互动挑战赛。系统将用户运动数据加密存储于云服务器。

1. 下列关于该信息系统中数据与信息的叙述，正确的是 C
   A. 该系统各类数据的呈现形式相同
   B. 用户运动数据加密存储会降低其价值
   C. 通过手机 APP 查看空气质量体现了信息具有共享性
   D. 语音紧急呼叫时无需对语音进行数字化
2. 下列关于该信息系统中硬件与软件的说法，正确的是 D
   A. 该系统中的传感器属于输出设备
   B. 该手机 APP 属于系统软件 应用软件
   C. 该系统云服务器中不需要安装操作系统
   D. 云服务器的存储容量对系统性能可能会产生影响
3. 下列关于该信息系统功能与安全的叙述，正确的是 A
   A. 该系统的数据分析功能主要由云服务器实现
   B. 运动数据存储在云服务器中，不存在数据丢失风险
   C. 市民通过手机 APP 参与线上健身挑战赛没有隐私泄露风险
   D. 查询健身互动挑战赛结果仅涉及系统的查询功能
4. 下列关于该信息系统通过深度学习实现 AI 人流量统计的说法，正确的是 C
   A. 需在灯杆本地存储所有训练样本
   B. 识别结果不受摄像头拍摄角度影响
   C. 可通过算法优化减少传输数据量
   D. 该功能属于行为主义人工智能 联结主义

阅读下列材料，回答第 5 至 6 题。

上述智慧健身步道系统中的灯杆包含普通和智能两种类型，每根灯杆具有唯一标识码。标识码结构为：区域号（4 位二进制） + 类型码（1 位二进制，0 表示普通灯杆，1 表示智能灯杆）+序号，其中序号的二进制位数由灯杆数量确定。灯杆标识码通过因特网传输至云服务器。云服务器接收到灯杆标识码后，根据预定义的标识码结构进行分解并获取其含义。

4位序号　3位序号　区域号4位+类型1位+序号4=9

5．若该区域有 12 根普通灯杆，5 根智能灯杆，则标识码所需的二进制位数最少是 D

A．3　　B．4　　C．8　　D．9

6．某灯杆标识码数据传输到云服务器的过程中，下列说法正确的 B

A．传输标识码数据不涉及网络协议

B．为保证数据传输的完整性，传输过程一般需包含数据校验机制

C．标识码传输至云服务器的过程中，不需要经过网关

D．标识码传输至云服务器所需时长仅由传输距离确定　网速、数据量大小等

7．某城市出租汽车计费方法：

白天：起步价 10 元（3 千米以内）；里程超出 3 千米（含）且在 15 千米以内时，超出 3 千米的部分，每千米按 2 元计费；里程超出 15 千米时（含），15 千米以外的部分，每千米按 3 元计费。

夜间：起步价 11 元（3 千米以内），其他计费方式同白天。

该计费流程图如第 7 题图所示，其中输入 f 为 0 或 1，分别表示白天和夜间，（1）～（4）可选表达式为

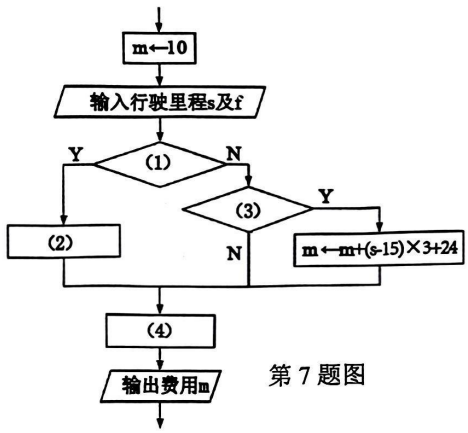
①m←m + f　　②m←m + (s - 3)×2

③3≤s<15?　　④s≥15?

则（1）～（4）处表达式序号依次为 B

A．③①④②　　B．③②④①

C．④①③②　　D．④②③①



第 7 题图

8．某非循环队列初始状态图如第 8 题图 a 所示，队列总长度 9，head 和 tail 分别表示队首和队尾指针。元素出队后直接输出或重新入队。经过若干步操作后，该队列状态如第 8 题图 b 所示，则当所有元素出队时，可能的输出结果为 B

第 8 题图 a

第 8 题图 b

A．DCAB　　B．CDAB

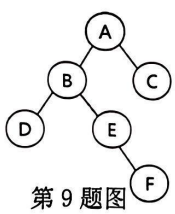
C．CBAD　　D．BACD

9．某二叉树如第 9 题图所示，每一个节点在中序遍历中都有一个位置序号。若删除叶子节点 F，则下列选项中的节点，在中序遍历中的位置序号相比删除前发生变化的是 A

A．节点 A　　B．节点 B

C．节点 D　　D．节点 E

删除前中序：DBEFAC
删除后中序：DBEAC



第 9 题图

10．编写如下所示甲、乙 Python 程序段：

甲程序段	乙程序段
k=0 while k<int(len(a)/2): 　　a[k],a[len(a)-k-1]=a[len(a)-k-1],a[k] 　　k+=1　　两头数据交换	x=0;y=len(a)-1 while x<y: 　　a[x],a[y]=a[y],a[x] 　　x+=1;y-=1

下列对甲、乙程序段的描述，不正确的是 D

A. 甲、乙程序段的时间复杂度均为 O（n）
B. 若列表 a 的值为[1,2,3,4,5]，分别执行甲、乙程序段，列表 a 的运行结果一致
C. 将甲程序段循环条件中的“<”改为“<=”，修改前后的程序段功能不一致
D. 对任意列表 a，甲、乙程序段分别运行结束后，x+y 的值为 k 的两倍

多了等号，奇数个时，中间位自己和自己换

11. 定义如下函数：

```
def bbsort(data):
    blk = []
    st = 0
    for i in range(1, len(data)):
        if data[i] != data[i - 1]:
            ed = i - 1
            blk.append([st, ed])
            st = i
    n = len(blk)
    for i in range(n):
        for j in range(0, n - i - 1):
            if data[blk[j][0]] > data[blk[j + 1][0]]:
                blk[j], blk[j + 1] = blk[j + 1], blk[j]
```

0 1 2 3
y x k=2

记录重复区间的开始和结束下标
注意：碰到不相等的数值时，blk才添加区间，所以data[12]没有被添区间
blk=[[0, 2], [3, 4], [5, 8], [9, 10]]

根据每个区间的开始位置的数值排序
[[3, 4], [0, 2], [5, 8], [9, 10]]

调用该函数，若参数 data 为[2, 2, 2, 1, 1, 3, 3, 3, 3, 7, 7, 7, 4]，运行该程序段后，blk[x][1]（0≤x≤len(blk)-1）的值不可能为 D

A. 2　　B. 4　　C. 11　　D. 12

12. 有如下 Python 程序段：

```
lnk = [[1, -1]]
head = 0
for i in range(len(a)):
    p = head
    while p != -1:
        if a[i][0] == lnk[p][0]:
            break
        q = p
        p = lnk[p][1]
    if ____(1)____:
        lnk.append(____(2)____)
        if p == head:
            head = len(lnk)-1
        else:
            lnk[q][1] = len(lnk)-1
    else:
        lnk.append(____(3)____)
        lnk[p][1] = len(lnk)-1
```

a[i][1]==0

i 0 1 2 3
[1,0],[2,1],[1,0],[1,1]
变成 2 3 4 5
推导 i+2

插入在q之后、p之前

插入在p之后

若 a 为[[1,0],[2,1],[1,0],[1,1]]，运行程序段后，lnk 为[[1, 4], [2, 2], [3, 3], [4, 0], [5, -1]]。程序段中(1)～(3)处可选语句为：

①a[i][1] == 0　　②a[i][1] == 1
③[i+1, lnk[p][1]]　　④[i+2, lnk[p][1]]
⑤[i+1, p]　　⑥[i+2, p]

2->3->4->1->5

5在1之后，说明a[i][1]为1时插在p之后

则(1)～(3)处语句序号依次为 C

A. ②④⑤　　B. ①⑤③　　C. ①⑥④　　D. ②③⑥

则(1)～(3)处语句序号依次为 C

A. ②④⑤　　B. ①⑤③　　C. ①⑥④　　D. ②③⑥

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

13. 某农业基地搭建灌溉监测系统，对基地中的 6 块种植区域（编号为 0 到 5）进行土壤湿度监测和灌溉。监测与灌溉规则是：系统每隔 1 小时通过土壤湿度传感器依次采集各区域土壤湿度数据（湿度值在 0 到 100 范围内），当土壤湿度低于阈值（湿度值 30）时启动灌溉设备（每个区域有灌溉设备，每次灌溉 2 小时），若所有区域土壤湿度均不低于阈值时，对湿度最低的区域进行灌溉（若有多个区域满足条件，均灌溉）；正在灌溉中的区域不进行土壤湿度采集和监测。请回答下列问题：

（1）若某次采集到 0～5 号区域土壤湿度依次为 45，38，38，42，47，42，当时没有区域正在灌溉，则需要启动灌溉设备的种植区域编号为 1,▲2 （填写区域编号，若有多个，中间用逗号分隔）。

（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。

```
n = 6                  # 种植区域数量
h = [0] * n            # 依次存储采集到的各区域湿度数据
flag = [False] * n     # 灌溉状态标记
start_t = [-1] * n     # 记录开始灌溉时间（小时）
ctime = 0              # 系统运行时长（小时），系统刚启动时记为 0
while True:
    # 采集各区域湿度数据存入 h，h[i]存放 i 号区域的湿度值，代码略
    t=[]    ①
    min_h = 101
    for i in range(n):
        if not flag[i] and h[i] < 30:
            t.append(i)
    if len(t) == 0:
        for i in range(n):
            if    ②    :
                if h[i] < min_h:
                    t = []
                    min_h = h[i]
                t.append(i)
    # 执行灌溉并更新状态
    for i in t:
        # 对 i 号区域进行灌溉，代码略
        flag[i] = True
        ③    start_t[i] = ctime
    # 检查灌溉结束条件（2 小时周期）
    for i in range(n):
        if flag[i] and ctime - start_t[i] >= 2:
            flag[i] = False
    # 延时 1 小时，代码略
    ctime += 1
```

not flag[i] and h[i] <=min_h

h[i]==min_h时，t也要添加i

14. 某市环保部门搭建了城市噪声监测系统，在 n 个重点区域各设置 1 个监测点。每个监测点通过一个智能终端连接一个噪声传感器，每隔 15 分钟采集一次噪声数据（单位：分贝），通过 5G 模块将数据传输到 Web 服务器（IP 地址为 47.111.143.0），并存储到数据库中。当服务器判定噪声值超过 85 分贝时，通过智能终端控制该区域警示设备发出警示信息（红色：噪声超标；绿色：噪声未超标），管理员通过浏览器查看各区域噪声信息。请回答下列问题：

（1）管理员通过巡查发现，某一区域实际噪声值持续超过 85 分贝，通过浏览器查看到的噪声监测值大于 85 分贝，但警示设备没有发出警示信息，以下情况可能导致该现象发生的是____▲____（单选，填字母）。

C

A. 该区域智能终端与 Web 服务器连接故障

B. 该区域噪声传感器与智能终端连接故障

C. 该区域警示设备与智能终端连接故障

（2）基于 Flask Web 框架编写的部分服务器端代码如下。若某区域传感器编号为 6，采集的噪声值为 85，则智能终端在传输数据时使用的 URL 是 http://____▲____。

47.111.143.0:5000/input?val=85&id=6

```
# 导入 Falsk 框架模块及其他相关模块，代码略
app = Flask(__name__)
@app.route('/input',methods=['GET'])# 采用 GET 方式提交数据
def index():
    # 获取某区域传感器编号 id 和噪声 val 的值，存入数据库，代码略
# 服务器其他功能，代码略
if __name__ == '__main__':
    app.run(host = '0.0.0.0', port = 5000)
```

（3）下列关于该系统设计及使用的说法，正确的有____▲____（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）

AC

A. 同一监测点的警示设备和噪声传感器可以连接在同一个智能终端

B. 某区域噪声传感器出现故障，会影响其他区域噪声传感器正常工作

C. 考虑到该系统数据量不大，可以采用 SQLite 3 数据库

D. 考虑到城市范围，将 5G 模块改用蓝牙上传数据到服务器更合适

蓝牙距离一般10米

（4）为使得某区域的市民更好了解该区域的噪声污染程度，并提升系统警示准确度，需要修改系统设计。请针对污染程度和警示准确度分别给出一种合理修改建议。

针对污染程度：根据检测到的噪声值大小来调整警示设备的警示颜色种类、持续时长等；

针对污染警示准确度：减小该区域智能终端采集噪声传感器的时间间隔

（5）管理员整理出 6 月份各区域噪声监测值并存入数据文件“noise_data.xlsx”，部分数据如第 14 题图 a 所示，现要找出 6 月份噪声超过阈值（85 分贝）总次数最多的监测点（数据保证只有一个），并输出该监测点超过阈值次数最多的 3 天（如果并列，取日期靠前的），运行结果如第 14 题图 b 所示。

日期	时间	监测点	实测噪声
2025-06-01	00:00:00	A区	67.2
2025-06-01	00:15:00	A区	70.2
[illegible]			
2025-06-30	23:15:00	E区	59.1
2025-06-30	23:30:00	E区	55
2025-06-30	23:45:00	E区	58.5

第 14 题图 a

```
6月份超过阈值总次数最多为：A区
该区超过阈值次数最多的3天为：
2025-06-10 14 次
2025-06-01 13 次
2025-06-02 13 次
```

第 14 题图 b

实现上述功能的部分 Python 程序如下：

```
import pandas as pd
df = pd.read_excel("noise_data.xlsx")
```

①A
②C

```
# ignore_index=True 表示产生新的索引，索引值默认为从 0 开始递增的整数
df1 = df1.sort_values("时间", ascending=False, ignore_index=True)
print("6 月份超过阈值总次数最多为:", df1.at[0, "监测点"])
```

③B
④D

```
df1 = df1.sort_values("时间", ascending=False)          # 降序排序
df1 = df1.head(3)
# 输出该区超过阈值次数最多的 3 天信息，代码略
```

程序中①～④处应填入的语句依次为__▲__（填字母，次序错不得分）。

```
A. df = df[df["实测噪声"] > 85]                          # 筛选数据
B. df1 = df[df["监测点"] == df1.at[0, "监测点"]]         # 筛选数据
C. df1 = df.groupby("监测点", as_index=False).count()    # 分组计数
D. df1 = df1.groupby("日期", as_index=False).count()     # 分组计数
```

15. 现需要对依次输入的一批整型数据进行存储，要求不重复，且存储后的数据保持升序有序。为此，小明设计如下数据存储方法：

一开始数据存储区域为空（程序实现时用空列表[]表示），当某个输入数据需要存储时，先判断相同的数据是否已经存储过，若已存储，则不再重复存储，否则将该数据存入列表，并保证列表中的数据升序有序。

如现在需要依次存储的数据为 13，7，25，13，7，24，……，存储数据的过程如第 15 题图所示。

操作序号	待存储数据	存储结果
1	13	[13]
2	7	[7,13]
3	25	[7,13,25]
4	13	[7,13,25]
5	7	
…	…	…

7已经出现过，不在重复存储
[7,13,25]

第 15 题图

随着存储过程的进行，列表中的数据越来越多，为提高数据存储效率，小明对数据存储方法进行了优化：先将已经存储数据的列表依次分成若干块（每块元素个数为 m，最后一块元素若不足 m 个时，单独成一块），然后在分块数据的基础上再进行后续的数据存储。

请回答下列问题：

(1) 根据小明设计的数据存储方法，执行第 15 题图所示操作序号为 5 的操作（即待存储数据 7）后，存储结果中的第 3 个元素为25▲__。

(2) 定义如下 block_partition(a, m)函数，参数 a 为按升序排序后的列表（元素均为整型）。函数功能是将列表 a 从头至尾依次分成若干块（每块 m 个元素），最后一块元素若不足 m 个时，单独成一块，并记录每块的起始和结束位置索引。

```
def block_partition(a, m):
    blocks = []
    n = len(a)
    start = 0
    while start < n:
        end = min(start + m - 1, n) # 函数 min(x,y)求 x 和 y 的较小值
        blocks.append([start, end])
        start = end + 1  # 下一块起始索引
    return blocks
```

一个区块满m个，结束位置是start+m-1
最后一个区块可能不满m个，以a数组的最后一个下标为准

min(start + m -1, n -1)

①调用 block_partition 函数，若列表 a 为[1, 2, 5, 7, 9]，m 为 2，则返回结果 blocks 中的第一个元素为[0,1]，最后一个元素为__▲__ [4,4]

②程序中加框处代码有错，请改正。bolcks=[[0,1],[2,3],[4,4]]

(3) 实现优化后数据存储功能的 Python 程序段如下，请在划线处填入合适的代码。

```
def insert_element(a, blocks, key, m, bi):#存储数据 key 到列表 a 中，并更新块信息
    if len(a) == 0:
        a.append(key)
        blocks = [[0, 0]]
        return
    if bi == -1:
        a.append(key)
    else:
        left, right = blocks[bi][0], blocks[bi][1]
        while left <= right:
            mid = (left + right) // 2
            if a[mid] < key:
                left = mid + 1
            else:
                right = mid - 1
        ①_____
        a.insert(pos, key) # 在列表 a 的 pos 索引位置插入数据 key
    # 更新块信息
    lb = len(blocks) - 1
    if blocks[lb][1]-blocks[lb][0] + 1 < m:
        blocks[lb][1] = len(a) - 1
    else:
        blocks.insert(_____②_____) # 创建新的块
```

若a为空列，添加key

此时key>a[end],所以将key插入在最后

在a中间用对分查找的思想找key插入的位置

停止线法

right | left

① pos=left 或pos=right+1

lb + 1,[len(a) -1,len(a) -1]

insert(插入位置，插入内容)

或lb + 1,[blocks[lb][1]]+1,blocks[lb][1]]+1]
或lb + 1,[blocks[lb][0]]+m,blocks[lb][0]]+m]

```
def find_element(a, blocks, key):
    for i in range(len(blocks)):
        start, end = blocks[i][0], blocks[i][1]
        if key <= a[end]:
            left, right = start, end
            while left <= right:
                mid = (left + right) // 2
                if a[mid] == key:
                    return True
                elif a[mid] < key:
                    left = mid + 1
                else:
                    right = mid - 1
            insert_element(a, blocks, key, m, i) # 插入
            return False
    return False
'''
算法优化前，部分数据已存入列表 a
读入分块大小，存入 m，代码略
'''
blocks = block_partition(a, m)        #对列表 a 分块
'''
继续依次存储数据，对每个待存储数据 key 调用 find_element(a, blocks, key)
实现存储
代码略
'''
def insert_element(a, blocks, key, m, bi):#存储数据 key 到列表 a 中，并更新块信息
    if len(a) == 0:
        a.append(key)
        blocks = [[0, 0]]
        return
    if bi == -1:
        a.append(key)
    else:
```

用二份查找的方式在各个区块中查找key是否存在，若存在则插入在该区块中，如果

insert_element(a,blocks,key,m,-1)

若a为空列，添加key

此时key>a[end],所以将key插入在最后

在a中间用对分查找的思想找key插入的位置