绝密★考试结束前

浙江省普通高校招生选考科目模拟考试

技术　试题

姓名：_____　　准考证号：_____

考生须知：

1．本卷满分 100 分，考试时间 90 分钟；

2．答题前，在答题卷指定区域填写班级、姓名、考场、座位号及准考证号并核对条形码信息；

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、错选、多选均不得分。）

1.下列关于数据、信息和知识的说法，正确的是（　　）

A 在数据处理过程中不会有新的信息产生

B 只要获取足够的信息，就能掌握丰富的知识

C 信息在数字化后才可以进行传递和共享

D 数据是信息的载体，数据经过解释可以获得信息

2.下列关于人工智能的说法，正确的是（　　）

A 行为主义人工智能通过模仿人类大脑中神经元之间的复杂交互来进行认知推理

B 强化学习应用了行为主义人工智能方法

C 网络上获取的数据可直接应用于深度学习

D 联结主义人工智能依据“交互—反馈”进行学习

阅读以下材料回答第 3 到 5 题

某超市配备了“智能购物车”，用户只需要将物品扫码后放入购物车，购物车就会自动计费，同时还具备商品推荐和导航功能。该系统的主要硬件参数及软件功能如下：

硬件参数	软件功能
智能终端：搭载鸿蒙操作系统，具备无线 WIFI 连接功能，搭配 10.1 寸显示屏 摄像头：前置及后置全高清摄像头 搭载传感器：陀螺仪（方向传感）、重力感应器、射频感应器等多款传感器	超市购物系统：刷脸登录、商品扫码、商品称重及金额计算、刷脸或扫码支付 商品推荐系统：根据过往消费者喜好，推荐优惠商品 路径导航系统：根据用户需求提供路径导航

3.下列关于该信息系统的说法，正确的是（　　）

A 智能终端上既要安装系统软件，也要安装应用软件

B 网络环境的好坏，不会对智能购物车的使用造成影响

C 摄像头用于刷脸及商品扫描，属于输出设备

D 购物车可以直接拿到其他超市去使用

4.下列关于该信息系统中使用的软硬件技术的说法，不正确的是（　　）

A 根据消费者需求提供路径导航，依靠的是陀螺仪和导航技术

B 找到商品二维码添加到购物车，依靠的是摄像头和文字识别技术

C 用户登录后根据喜好推荐商品，依靠的是人脸识别和大数据处理技术

D 商品称重及自动金额计算，依靠的是重力传感器和购物系统

5.下列关于该信息系统中数据的说法不正确的是（　　）

A 用户扫码商品的数据，可以保存在购物车的智能终端上

B 用户登录购物系统后，以往的消费数据会从系统数据库中被提取并分析

C 用户的购买记录，在用户付款结算后会保存到系统数据库中

D 用户增删当前购物车的商品，必须实时将数据上传到系统数据库

6. 以下关于网络的一些说法，正确的是（　　）

A 有线接入网络的设备需要 IP 地址，无线接入的不需要

B 在同一局域网中，不同的大楼之间允许出现相同 IP 地址的终端

C 在局域网中，DHCP 服务器负责将局域网的私有地址转换为公有地址

D 网址一般称作 URL，由网络协议、服务器地址及文件名等组成

7. a、b 是两个正整数，a 不能被 b 整除。下列选项中，表达式结果与其他三项不同的是（　　）

A　int(a/b)==a/b　　B　a-a/b*b==0

C　a//b==a/b　　D　a%b==0

8.设栈 S 和队列 Q 的初始状态为空，元素按 a1，a2，a3，a4，a5，a6 的顺序依次通过栈 S 后进入队列 Q。若队列 Q 中的元素依次为 a2，a4，a3，a6，a5，a1，则栈 S 的容量至少是（　　）

A 6　　B 4　　C 3　　D 2

9.假设完全二叉树的树根为第 1 层，树中第 10 层有 5 个叶子节点，则完全二叉树最多有多少个节点？（　　）

A 2047　　B 2048　　C 2037　　D 2038

10.有如下 python 程序段：

```
m=int(input())
def dex_oct(m):
    if m>=8:
        dex_oct(m//8)
        print(m%8,end='')
    else:
        print(m,end='')
dex_oct(m)
```

运行该程序时，如果输入 15，则输出的结果是（　　）

A　81　　B　71　　C　17　　D　18

11.有如下 Python 代码：

```
import random
a=[25,56,74,28,40,34]
b=[0,1,2,3,4,5]
n=len(a)
x=random.randint(0,2)
for i in range(n-1-x):
    for j in range(n-1,i,-1):
        if a[b[j]]>a[b[j-1]]:
            b[j],b[j-1]=b[j-1],b[j]
```

程序执行后列表 b 的值不可能的是（　　）

A　[2,1,4,5,3,0]　　B　[2,1,4,5,0,3]　　C　[2,1,4,0,5,3]　　D　[2,1,0,4,5,3]

12.有一个数组，它的偶数位是一个升序的奇数，奇数位是降序的偶数，如[1,10,3,8,5,6,7,4,9,2]

因原数组中元素太多，小明想用对分查找的方法来确定各个元素所在的位置，请填充以下程序：

```
#自定义函数 df，参数 a 是有序列表，k 是要查找的数，返回值为 k 所在的位置,找不到返回-1
def df(a,k):
    i=0
```

```
    j=len(a)-1
    while i<=j:
        m=(i+j)//2
        if a[m]==k:
            return m
        elif a[m]>k:
            j=m-1
        else:
            i=m+1
    return -1
k=int(input('请输入 k：'))
a=[1,10,3,8,5,6,7,4,9,2]
b=[];c=[]
for x in range(len(a)):
    if x%2==0:
        b.append(_____)
    else:
        c.append(_____)
if k%2==1:
    y=_____
else:
    y=_____
if y>=0:
    print(y)
else:
  print(‘k 不在列表中’)
```

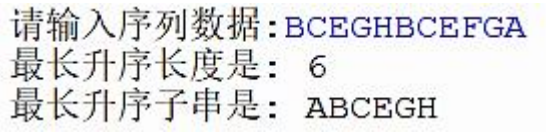
划线处填入的代码可以从下面的代码中选择①a[x]　②-a[x]　③df(b,k)*2　④df(c,-k)*2+1

则正确选项为（　　）

A　①②③④　　B　②①③④　　C　①②④③　　D　②①④③

二、非选择题（本大题共 3 小题，其中第 13 小题 7 分，第 14 小题 11 分，第 15 小题 8 分，共 26 分）

13. 小杜是一名数据分析师，正在开发一个处理特定序列数据的应用程序，这些数据是以环状序列的形式提供的。现在需要设计一个算法，能够在这些环状序列中找出最长升序子串(若有多组最长子串，则输出最先出现那组)。例如环状序列“BCEGHBCEFGA”中最长升序子串是“ABCEGH”。程序运行界面如第 13 题图所示。



第 13 题图

实现上述功能的程序段如下，请回答下列问题：

(1)请在划线处填入合适的代码。

(2)加框处代码有误，请改正。

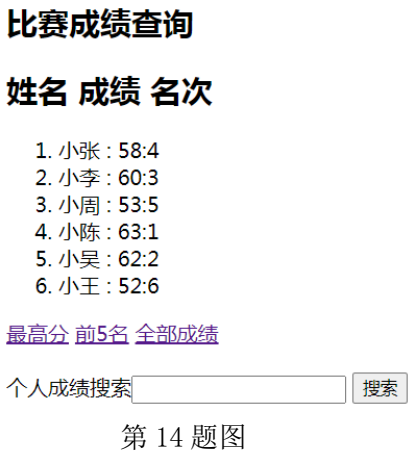
```
s=input("请输入序列数据:")
n=len(s)
for i in range(n-1):
    if s[i-1]>s[i] :
        break
```

```
    if i==n-2:
        k=n-1
        maxc=n
    else:
        maxc=1
        cnt=1
        p1=i+1
        p2=(p1+1)%n
        while ____①____:
            if s[p1]<s[p2]:
                cnt+=1
                if cnt>maxc:
                    maxc=cnt
                    ____②____
            else:
                cnt=1
            p1=p2
            p2=(p2+1)%n
    print("最长升序长度是:",maxc)
    s1=""
    for i in range(maxc):
        s1=s[k]+s1
        ____③____
    print("最长升序子串是:",s1)
```

14.某比赛组委会为了工作方便，决定开发一个成绩发布系统。工作人员在系统中录入成绩后，选手和观众可以在浏览器中查看各选手的成绩，如第14题图：

部分代码如下：

```
import pandas as pd
from flask import Flask, render_template,request
app=Flask(__name__)
@app.route('/',methods=["GET","POST"])
def quanbancj():
    if request.method=="GET":
        return render_template('index.html', cj=df.values)
    else:
        xingm=request.form.get("xm")
        for i in a:
            if xingm==i[0]:
                return xingm+':'+str(i[1])
        else:
            return "没有人叫  "+xingm
@app.route('/q1')
def zuigaofen():
    px=df.sort_values('成绩')
    zgf=px.tail(1)
    return render_template('index.html', cj=zgf.values)
```



第 14 题图

```
@app.route('/q5')
def qian5min():
    px=df.sort_values('成绩')
    zgf=px.tail(5)
    return render_template('index.html', cj=zgf.values)
app.run(host='0.0.0.0',port=80)
```

(1) 组委会在决定采用网站发布选手的比赛成绩，在搭建信息系统的前期准备工作中属于_____阶段（单选，填字母：A.需求分析　B.可行性分析　C.概要设计　D.详细设计），采用了_____架构（B/S 或 C/S）。

(2) 如果服务器所在的电脑的 IP 地址为 172.28.12.30，那么要查看本次比赛前 5 名的情况，应该在浏览器中输入的网址为_____

(3) 由于比赛人员多，网页一面显示不完，某选手通过搜索框搜索他自己，请问单击搜索按钮时所使用的请求是_____（GET/POST）

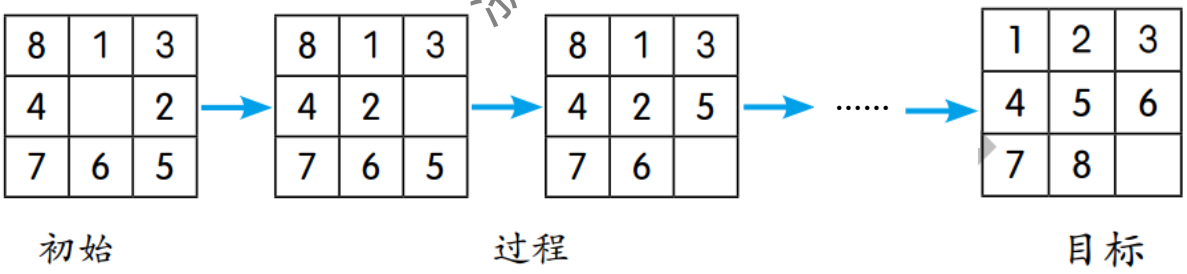
(4) 该信息系统投入使用过程中，好多用户反映网页打开的速度慢，请写出 2 种提升用户体验的改进方法：

_____

(5) 某次比赛的成绩为 [["小张",58],["小李",60],["小周",53],["小陈",63],["小吴",62],["小王",52]]，为求出各选手的名次，请完善下面的代码：

```
df=pd.DataFrame(a,columns=['姓名','成绩'])
df['名次']=0
for i in range(len(df)):
    ①_____
    for j in range(len(df)):
        if df.at[j,'成绩']>=df.at[i,'成绩']:
            c+=1
    ②_____
```

15.有一个经典的小游戏，主要任务是把一个九宫格内排乱的八个数码格利用空格，按要求重新排列好。移动时，只能横向或纵向将空格与其他格子交换位置。具体可参见第 15 题图。



第 15 题图

小明编写了一段 Python 程序来模拟该游戏，在测试过程中发现并不是每一种初始的八数码排列布局，最后都能转换为目标布局，在长期的研究过程中，发现可以通过计算逆序数的方法来判断八数码问题是否有解，如果初始序列与目标序列的逆序数的奇偶性相同，则问题有解，否则无解。

（1）#计算序列对应的逆序数

```
def inverse(data):
    ns=0
    for i in range(1,9):
        if data[i]==0
            continue
        for j in range(0,i):
            if ____①_____:
```

```
                ns=ns+1
    return ns
```

（2）在移动格子的过程中，要是每一步都跟最后的结果靠近一点，一步步走下来，自然就可以解决问题了。但问题在于，怎么知道一个排列方案比另一个排列方案更接近目标，我们可以使用差异评估法，它分为两部分：①统计有多少个数字格不在自己的位置上，结果记为 *f(n)*； ②所有位置不对的数字格，其横、纵坐标与目标位置横、纵坐标的差的绝对值，结果记为 *h(n)*。 当一个方案的 *f(n)+h(n)*比另一方案小的时候，就认为这个方案更接近目标。

```
def comp_differ(data):    # 评估差异度
    f, h = [], []
    for index, n in enumerate(data):   #enumerate 函数把 data 列表的各元素索引给 index，各元素的值给 n
        if n == 0:
            continue
        if index == n - 1
            f.append(0)
        else:
            f.append(1)
        cx = index // 3
        ②
        x, y = ps[n - 1]
        h.append(abs(cx - x) + abs(cy - y))
    return sum(f) + sum(h)
def get_steps(data): #根据差异评估值，筛选出差异度更小的排列方案
    i0 = data.index(0)
    ds = [data[ci] for ci in cr[i0]]
    r, dif = None, 100        # 默认交换的格子为 None，默认差异度是 100

    for d in ds:
        t, id = data, data.index(d)
        t[i0], t[id] = t[id], t[i0]
        if t in puzs:
            continue
        ③                                    #评估差异度
        if td < dif:
            dif = td
            r = t
    return r
ps={0:(0,0),1:(0,1),2:(0,2),3:(1,0),4:(1,1),5:(1,2),6:(2,0),7:(2,1),8:(2,2)}
cr={0:(1,3),1:(0,2,4),2:(1,5),3:(0,4,6),4:(1,3,5,7),5:(2,4,8),6:(3,7),7:(4,6,8),8: (5,7)}  # 可移动位置关系
steps, puzs = [], []
def solve(data):
    result=False
    while data != mubiao:
        steps.append(data)
        if data not in puzs:
```

```
            puzs.append(data)
        data = get_steps(data)
        if data == None:
            steps.pop()
            data = steps[-1]
        if len(steps) >  50000:
            break       # 步数太多就强制退出
    else :
        result=True
    return result
a = [8,1,3,4,0,2,7,5,6]
mubiao = [1,2,3,4,5,6,7,8,0]
if _____④_____:
    print('无解')
else:
    if solve(a) :
        print(f'解出来了，共{len(steps)}步。')
    else :
        print('失败了!')
```