GESP Python 五级样题卷

(满分：100 分　考试时间：90 分钟)

学校：_____　姓名：_____

题目	一	二	三	总分
得分				

一、单选题（每题 2 分，共 30 分）

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	A	C	B	C	A	D	D	A	D	B	C	D	C	C	B

1． 以下不属于计算机输出设备的有（　）。

A． 麦克风

B． 音箱

C． 打印机

D． 显示器

2． 关于分治算法，下列说法错误的是（　）。

A． 分治算法的核心思想是分而治之，即把问题转化为多个规模更小的子问题求解

B． 分治算法可以不使用递归实现

C． 分治算法的时间复杂度是 O(logN)，其中 N 表示问题的规模

D． 二分法、快速排序等算法都是典型的使用分治思想的算法

3． 小明想了一个 1~100 之间的整数。你可以做多次猜测，每次猜测之后，如果你没有猜中，小明会告诉你，你猜的数比他想的数大还是小。你希望你在运气最坏的情况下花费最少的次数猜

中，请问你运气最坏的情况下会猜（　）次？（包括最后猜中的那次）

A. 5

B. 6

C. 7

D. 100

4. 有关下面 Python 代码说法错误的是（　）。

```
# factA 和 factB 用于求非负整数 n 的阶乘
def factA(n):
    if n <= 1:
        return 1
    ret, i = 1, 2
    while i <= n:
        ret *= i
        i += 1
    return ret


def factB(n):
    print(n)
    return 1 if n == 1 else n * factB(n-1)


N = int(input("请输入大于等于 1 的正整数:"))
print(factA(N), factB(N))
```

A. factA()采用循环方式求 n 的阶乘，factB()采用递归方式求 n 的阶乘

B. 倒数第 2 行被执行时如果输入 5，能实现求其阶乘，程序输出结果为 120 120

C. 任何递归程序都可以使用循环改写

D. 一般说来，factA()的效率高于 factB()

5. 下面 Python 代码意在实现字符串反序的功能，关于这段代码，以下说法正确的是（　）。

```
#字符串反序
def sReverse(sIn):
```

```
    if len(sIn) <= 1:
        return sIn
    else:
        return sReverse(sIn[1:]) + sIn[:1]

print(sReverse("Hello"))
```

A. 这段代码可以正确实现字符串反序的功能，其输出为 olleH

B. 这段代码不能正确实现字符串反序的功能，其输出为 Hello

C. 这段代码不能正确实现字符串反序的功能，其输出为 HHHHH

D. 这段代码不能正确实现字符串反序的功能，其输出为 ooooo

6. 阅读下面 Python 实现的二分查找代码，下列说法中错误的是（ ）。

```
def binarySearch(lst,itm):
    def __binarySearch(lst, l, r, x):
        if r <= l:
            return -1

        mid = l + (r - l) // 2

        if lst[mid] == x:
            return mid
        elif lst[mid] > x:
            return __binarySearch(lst, l, mid-1, x)
        else:
            return __binarySearch(lst, mid, r, x)

    return __binarySearch(lst, 0, len(lst) - 1, itm)
```

A. 上面代码实现的二分查找，最少只需查找一次即可得到结果

B. 如果调用该函数在列表[2, 3, 4, 10, 12]中查找元素 0，则它实际被调用 3 次

C. 如果调用该函数在列表[2, 3, 4, 10, 12]中查找元素 3，则它实际被调用 3 次

D. 如果调用该函数在列表[2, 3, 4, 10, 12]中查找元素 10，则它实际被调用 3 次

7. 关于下面 Python 代码的说法中，错误的是（ ）。

```
lstA = [1, 2, 5, 4, 3]
lstA.sort()
```

```
print(lstA) # line 3
lstA.sort(reverse=True)
print(lstA) # line 5

lstB = ["Alice", "Sam", "Tony"]
lstB.sort(key=len)
print(lstB) # line 9

lstC = [1, 3, '2']
lstC.sort()
print(lstC) # line 13
```

A. 第 3 行能正确执行，输出结果为 `[1, 2, 3, 4, 5]`

B. 第 5 行能正确执行，输出结果为 `[5, 4, 3, 2, 1]`

C. 第 9 行能正确执行，输出结果为 `['Sam', 'Tony', 'Alice']`

D. 第 13 行能正确执行，因为 python 会自动将字符 ``'2'`` 转为 ASCII 码的数值排序，输出结果为 `[1, 3, '2']`

8. 有关下面 Python 代码错误的是（ ）。

```
def f1():
    print("f1")
    g1()

def g1():
    print("g1")

def f2():
    print("f2")
    def g2(): # line 10
        print("g2")
    g2() # line 12


f1() # line 15
f2()
```

A. 第 15 行不能正确执行，因为第三行调用了函数 g1()，第五行才定义函数 g1()

B. 若将第 12 行的 g2()调用移到第十行的 g2 定义之前，会引发异常

C. 程序能够正确执行，共输出 4 行，依次为 `f1`、`g1`、`f2`、`g2`

D. 若在程序最后增加一行代码：`f2.g2()`，会引发异常

9. 有关下面 Python 代码错误的是（　）。

```
pow = lambda x: x ** 2

def mul(N, M):
    return N*M

def op(Fx):
    return Fx

print(op(mul)(2, 5))
_pow = pow
print(_pow(2))
print(op(pow)(8))
```

A. op()函数定义正确

B. 倒数第 4 行代码将 mul 作为 op()参数，可以正确执行

C. 函数名称可赋给其他量，后者也可作为函数调用，因此倒数第 2、倒数第 3 行代码可以正确执行

D. 最后一行代码将函数 pow 作为 op()参数，将导致错误

10. 下面代码执行后的输出是（　）。

```
def fibonacci(n):
    print(n, end=",")
    if n == 1 or n == 2:
        return 1
    return fibonacci(n - 1) + fibonacci(n - 2)

print(fibonacci(5))
```

A. 5,4,3,2,1,2,1,5

B. 5,4,3,2,1,2,3,2,1,5

C. 5,4,4,3,2,1,3,2,1,5

D. 5,4,3,2,1,3,2,1,5

11. 函数 f 的作用是（　）。

```
def f(a, b):
    return a if b == 0 else f(b, a % b)
```

A. 求 a 和 b 的最大公共质因子

B. 求 a 和 b 的最小公共质因子

C. 求 a 和 b 的最大公约数

D. 求 a 和 b 的最小公倍数

12. 下面 Python 代码用于排序，下列说法中错误的是（ ）。

```
def sortA(listData):
    n = len(listData)
    for i in range(n):
        for j in range(0, n-i-1):
            if listData[j] > listData[j+1]:
                listData[j], listData[j+1] = listData[j+1], listData[j]
    return listData


def sortB(listData):
    if len(listData) <= 1:
        return listData

    middle = len(listData) // 2
    left = sortB(listData[:middle])
    right = sortB(listData[middle:])

    p, q = 0, 0
    ret = []

    while len(left) > p and len(right) > q:
        if left[p] <= right[q]:
            ret.append(left[p])
            p += 1
        else:
            ret.append(right[q])
            q += 1

    ret += left[p:]
    ret += right[q:]

    return ret
```

A. 两种排序算法的时间复杂度不同

B. 两种排序算法的空间复杂度一致

C. sortA 的时间复杂度在最好和最坏情况下都是 $O(N^{2})$

D. sortB 的平均时间复杂度、最好情况的时间复杂度都是 $O(\log N)$，最坏情况的时间复杂度是 $O(N^{2})$

13. 在第 12 题的 sortB 函数中，代码 left = sortB(listData[:middle])和 right = sortB(listData[middle:])明显涉及到的算法思想是（ ）。

A. 递推算法

B. 贪心算法

C. 分治

D. 二分搜索算法

14. 有关下面 Python 代码正确的是（ ）。

```
def f():
    return 4

def g():
    print(4)

lstA = [1, 2, 3, f] # line 7
print(lstA) # line 8
del f, g
print(lstA)
g()
```

A. 第七行代码将导致错误

B. 第八行代码若能正确执行，则输出 `[1, 2, 3, 4]`

C. 倒数第二行代码能正确执行

D. 最后一行代码能正确执行

15. 关于链表说法正确的是（ ）。

A. 存储空间需要事先分配

B. 在链表头部插入元素的时间复杂度是 $O(1)$

C. 循环链表使得任意一个结点都可以很方便地访问其前驱与后继

D. 从双向链表的任意一个节点出发，并不一定能够访问到所有其他节点

二、判断题（每题 2 分，共 20 分）

题号	1	2	3	4	5	6	7	8	9	10
答案	√	×	×	×	×	√	×	√	√	×

1. 计算机硬件主要包括运算器、控制器、存储器、输入设备和输出设备。

2. 唯一分解定理指的是分解质因数只有唯一的一种算法。

3. 埃氏筛法用于找出自然数 N 以内的所有质数，其时间复杂度为 $O(N\sqrt{N})$，因为判定一个数是否为质数的时间复杂度为 $O(N)$。

4. 贪心法的每一步都采取局部最优策略，因此必然能实现全局最优。

5. set 的 pop 方法用于删除集合中的指定元素。

6. 下面的 Python 代码执行后将输出[1, -2, 3, -4]。

```
lst = [1, 3, -2, -4]
lst.sort(key=lambda x: abs(x))
print(lst)
```

7. 字典（dict）项的键（key）可以是字典。

8. 下面 Python 代码可以运行结束，输出[1, 3]。

```
def sort_x(x):
    del x[1]
    x.sort()

x = [3, 2, 1]
```

```
sort_x(x)
print(x)
```

9. 选择排序和快速排序都是不稳定的。

10. 二分查找法可以应用于有序序列（如升序排序的整数数组），也可以应用于无序序列（如乱序的整数数组）。

三、编程题（每题 25 分，共 50 分）

题号	1	2
答案		

1、编程题：小杨的锻炼问题

时间限制：1.0s

内存限制：128.0MB

【问题描述】

小杨的班级里共有 N 名同学，每位同学都有各自的锻炼习惯。具体来说，第 i 位同学每隔 a_i 天就会进行一次锻炼（也就是说，每次锻炼会在上一次锻炼的 a_i 天后进行）。

某一天，班上的 N 名同学恰好都来进行了锻炼。他们对此兴奋不已，想要计算出下一次所有同学都来锻炼，至少要过多少天。但他们不会计算，你能帮帮他们吗？

【输入描述】

第一行一个整数 N，表示同学的数量。

第二行 N 个用空格隔开的正整数，依次为 $a_0, a_1, \cdots\cdots, a_{N-1}$。

【输出描述】

输出一个整数，表示下一次所有同学都来锻炼，至少要过多少天。

【特别提醒】

在常规程序中，输入、输出时提供提示是好习惯。但在本场考试中，由于系统限定，请不要在输入、输出中附带任何提示信息。

【样例输入 1】

```
3
1 2 3
```

【样例输出 1】

```
6
```

【样例解释 1】

第一位同学每天都锻炼；第二位同学每 2 天锻炼一次；第三位同学每 3 天锻炼一次。因此，6 天之后，三位同学都会进行锻炼。在此之前，第二位同学只会在第 2,4 天进行锻炼，第三位同学只会在第 3 天进行锻炼，他们都无法相遇。

【样例输入 2】

```
4
2 4 8 16
```

【样例输出 2】

```
16
```

【样例解释 2】

第四位同学每 16 天锻炼一次，而第 16 天后也恰好是前三位同学锻炼的日子。

【样例输入 3】

```
4
2 4 6 8
```

【样例输出 3】

```
24
```

【数据规模】

对于 20%的测试点，保证 N=2。

对于 50%的测试点，保证 N≤4。

对于所有测试点，保证 2≤N≤10，1≤a_i≤50。

【参考程序】

```
import math

def gcd(n, m):

    return math.gcd(n, m)

def lcm(n, m):

    return n * m // gcd(n, m)

def lcmList(lst):

    if lst == []:

        return 1

    if len(lst) == 1:

        return lst[0]

    return lcm(lst[0],lcmList(lst[1:]))

N = int(input())

lst = list(map(int,input().split(" ")))

print(lcmList(lst))
```

2、编程题：小杨的列队问题

时间限制：1.0s

内存限制：128.0MB

【问题描述】

小杨的班级里共有 N 名同学，学号从 0 至 N-1。

某节课上，老师要求同学们进行列队。具体来说，老师会依次点名 M 名同学，让他们加入队伍。每名新入队的同学需要先站到队伍末尾（刚开始队伍里一个人都没有，所以第一个入队的同学只需要站好即可），随后，整个队伍中的所有同学需要按身高从低到高重新排序（身高相同的同学之间的顺序任意）。

排队很容易，但重新排序难倒了同学们。稍加讨论后，他们发现可以通过交换位置的方法来实现排序。具体来说，他们可以让队伍中的两名同学交换位置，这样整个队伍的顺序就会发生变化，多经过这样的几次交换后，队伍的顺序就可以排号。

*例如：队伍中有 4 名同学，学号依次为 10, 17, 3, 25，我们可以令 3 号同学和 10 号同学交换位置，则交换后的队伍顺序变为 3, 17, 10, 25，这就是一次交换位置。

聪明的小杨想要知道：在老师每次点名一位新同学加入队伍后，在原有队伍的基础上，同学们最少要进行几次**交换位置**，才能完成老师按身高排序的要求。

【输入描述】

第一行一个整数 N，表示同学的数量。

第二行 N 个用空格隔开的正整数，依次表示学号为 0, 1, ……, N-1 的同学的身高（不超过 2,147,483,647）。

第三行一个整数 M，表示老师点名的数量。

接下来 M 行，依次描述 M 次点名：每行一个整数 x（0<=x<N），表示要求学号为 x 的同学加入队伍。保证该名同学此前不在队伍中。

对于所有的测试点，保证 1<=M<=N<=2,000。对于 50% 的测试点，保证所有同学的身高互不相同。

【输出描述】

输出 M 行，依次表示对于每次点名，同学们最少要进行几次**交换位置**，才能完成按身高排序的要求。

【特别提醒】

在常规程序中，输入、输出时提供提示是好习惯。但在本场考试中，由于系统限定，请不要在输入、输出中附带任何提示信息。

【样例输入 1】

```
5
170 165 168 160 175
4
0
3
2
1
```

【样例输出 1】

```
0
1
1
2
```

【样例解释 1】

初始时队伍为空，身高为 170 的 0 号同学加入队伍，不需要任何交换位置。

接着，身高为 160 的 3 号同学加入队伍的末尾，此时两位同学需要进行依次交换位置，才能保证身高更矮的 3 号同学排在身高更高的 0 号同学前面。

接着，身高为 168 的 2 号同学加入队伍的末尾，此时队伍中的同学学号（身高）依次为 3(160)，0(170)，2(168)，此时 2 号同学可以和 0 号同学进行一次交换位置，即可完成排序要求。

接着，身高为 165 的 1 号同学加入队伍的末尾，此时队伍中的同学学号（身高）依次为 3(160)，2(168)，0(170)，1(165)，此时可以令 1 号同学和 2 号同学进行一次交换位置，使队伍变为 3(160)，1(165)，0(170)，2(168)；随后再令 0 号同学和 2 号同学进行一次交换位置，使队伍变为 3(160)，1(165)，2(168)，0(170)，即可完成排序要求。

【样例输入 2】

```
4
20 20 20 10
4
0
1
2
3
```

【样例输出 2】

```
0
0
0
1
```

【样例解释 2】

前三位加入队伍的同学（0, 1, 2 号同学）身高都相同，不需要进行任何交换位置。最后加入队伍的 3 号同学身高最矮，需要和队头的 0 号同学交换位置，方可完成排序要求。

【参考程序】

```
n = int(input())

height = list(map(int, input().split(' ')))

m = int(input())

queue = []

for i in range(m):

    x = int(input())

    queue.append(height[x])

    ans = 0

    last = -1

    for j in range(i, 0, -1):

        if queue[j] >= queue[j - 1]:
```

```
            break
        queue[j], queue[j - 1] = queue[j - 1], queue[j]
        if queue[j] != last:
            ans += 1
        last = queue[j]
    print(ans)
```