CCF 编程能力等级认证
Grade Examination of Software Programming

Python 六级

2023 年 12 月

1 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	B	C	B	D	D	C	D	D	B	B	D	B	D	D	B

第 1 题 通讯卫星在通信网络系统中主要起到（）的作用。

- **A.** 信息过滤
- **B.** 信号中继
- **C.** 避免攻击
- **D.** 数据加密

第 2 题 小杨想编写一个判断任意输入的整数N是否为素数的程序，下面哪个方法不合适？（ ）

- **A.** 埃氏筛法
- **B.** 线性筛法
- **C.** 二分答案
- **D.** 枚举法

第 3 题

内排序有不同的类别，下面哪种排序算法和冒泡排序是同一类？（ ）

- **A.** 希尔排序
- **B.** 快速排序
- **C.** 堆排序
- **D.** 插入排序

第 4 题 关于Python类和对象的说法，错误的是(　　)。

- **A.** 在Python中，一切皆对象，即便是字面量如整数5等也是对象
- **B.** 在Python中，可以自定义新的类，并实例化为新的对象
- **C.** 在Python中，内置函数和自定义函数，都是类或者对象
- **D.** 在Python中，不可以在自定义函数中嵌套定义新的函数

第 5 题 有关下面Python代码的说法，正确的是(　　)。

```
class Point:
    def __init__(self,X,Y):
        self.x = X
        self.y = Y
class Rect:
    def __init__(self,lefttop_Point,rightbottom_Point):
        self.left_top = lefttop_Point
        self.right_bottom = rightbottom_Point
    def __contains__(self,xy):
        if self.left_top.x <= xy.x <= self.right_bottom.x \
               and self.right_bottom.y <= xy.y <= self.left_top.y:
            return True

        return False

rectA = Rect(Point(10,10),Point(20,5))
print(Point(15,8) in rectA)
```

- A. 第17行代码执行后将报错，因为 `Rect` 类没有定义 `in` 运算符
- B. 第16行代码将 `Point` 对象作为参数，将导致错误
- C. in是成员运算符，不适用于 `Rect` 类
- D. 由于 `Rect` 类定义了 `__contains__` 魔术方法，因此第17行代码能正确执行

第 6 题 有关下面Python代码的说法，正确的是(　　)。

```
class newClass(object):
    objCounter = 0
    def __init__(self):
        newClass.objCounter += 1

classA = newClass()
classB = newClass()
print(newClass.objCounter)
print(classA.objCounter)
```

- A. 第8行代码错误，第9行正确
- B. 第9行代码错误，第8行代码正确
- C. 第8、9两行代码都正确
- D. 第4行代码可修改为 `objCounter += 1`

第 7 题 有关下面Python代码的说法，错误的是(　　)。

```
class biTreeNode:
    def __init__(self, val=None, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class biTree(object):
    def __init__(self, root=None):
        self.root = root
```

- A. 上列Python代码适用于构造各种二叉树
- B. 代码 `Root = biTree(biTreeNode(5))` 构造二叉树的根节点
- C. 代码 `Root = biTree( )` 可以构造空二叉树，此时 `Root` 对象的 `root` 属性值为 `None`
- D. 代码 `Root = biTree(biTreeNode( ))` 可以构造空二叉树，此时 `Root` 对象的 `root` 属性为 `Node`

第 8 题 基于上题类的定义，有关下面Python代码的说法错误的是（　）。

```python
def Search(root, val):
    if root is None:
        return None

    if root.val == val:
        return root
    else:
        rtn = search(root.left, val)

    if rtn != None:
        return rtn

    return search(root.right, val)
```

- [ ] **A.** 代码中 `Search( )` 函数如果查找到查找值的节点，则返回该节点的对象
- [ ] **B.** 代码中 `Search( )` 函数先搜索左子树，如果搜索不到指定值，则搜索右子树
- [ ] **C.** 代码中 `Search( )` 函数采用递归方式实现二叉树节点的搜索
- [ ] **D.** 代码中 `Search( )` 函数采用动态规划方法实现二叉树节点的搜索

第 9 题 有关下面Python代码的说法正确的是（ ）。

```python
class Node:
    def __init__(self, Val, Prv=None, Nxt = None)
        self.Value = Val
        self.Prev = Prv
        self.Next = Nxt

firstNode = Node(10)
firstNode.Next = Node(100,firstNode)
firstNode.Next.Next = Node(111,firstNode.Next)
```

- [ ] **A.** 上述代码构成单向链表
- [ ] **B.** 上述代码构成双向链表
- [ ] **C.** 上述代码构成循环链表
- [ ] **D.** 上述代码构成指针链表

第 10 题 对 `hello world` 使用霍夫曼编码（Huffman Coding），最少 bit（比特）为（ ）。

- [ ] **A.** `4`
- [ ] **B.** `32`
- [ ] **C.** `64`
- [ ] **D.** `88`

第 11 题 下面的 `fiboA( )` 和 `fiboB( )` 两个函数分别实现斐波那契数列，该数列第1、第2项值为1，其余各项分别为前两项之和。下面有关说法错误的是（ ）。

```python
def fiboA(N):
    if N == 0:
        return 1
    if N == 1:
        return 1
    return fiboA(N - 1) + fiboA(N - 2)

def fiboB(N):
    dp = [-1] * (N + 1)
    dp[0] = 1
    dp[1] = 1
    for i in range(2, N + 1):
        dp[i] = dp[i - 1] + dp[i - 2]
    return dp[N]
```

- A. `fiboA( )` 采用递归方式实现斐波那契数列
- B. `fiboB( )` 采用动态规划算法实现斐波那契数列
- C. 当N值较大时，`fiboA( )` 存在大量重复计算
- D. 由于 `fiboA( )` 代码较短，其执行效率较高

第 12 题 有关下面Python代码不正确的说法是（ ）。

```python
def getDepth(root):
    if root is None:
        return 0
    left_depth = getDepth(root.left)
    right_depth = getDepth(root.right)
    if left_depth < right_depth:
        return right_depth + 1
    else:
        return left_depth + 1
```

- A. 该代码可用于求解二叉树的深度
- B. 代码中函数 `getDepth( )` 的参数 `root` 表示根节点，非根节点不可以作为参数
- C. 代码中函数 `getDepth( )` 采用了递归方法
- D. 代码中函数 `getDepth( )` 可用于求解各种形式的二叉树深度，要求该二叉树节点至少有 `left` 和 `right` 属性

第 13 题 下面有关树的存储，错误的是（ ）.

- A. 完全二叉树可以用 `list` 存储
- B. 一般二叉树都可以用 `list` 存储，空子树位置可以用 `None` 表示
- C. 满二叉树可以用 `list` 存储
- D. 树数据结构，都可以用 `list` 存储

第 14 题 下面有关Python中 `in` 运算符的时间复杂度的说法，错误的是（ ）。

- A. 当 `in` 运算符作用于 `dict` 时，其时间复杂度为$O(1)$
- B. 当 `in` 运算符作用于 `set` 时，其时间复杂度为$O(1)$
- C. 当 `in` 运算符作用于 `list` 时，其时间复杂度为$O(N)$
- D. 当 `in` 运算符作用于 `str` 时，其时间复杂度为$O(N)$

第 15 题 下面有关 `bool()` 函数的说法，正确的是（ ）。

- ☐ **A.** 如果自定义类中没有定义魔术方法 `__bool__( )`，将不能对该类的对象使用 `bool( )` 函数
- ☐ **B.** 如果自定义类中没有定义魔术方法 `__bool__( )`，将查找有无魔术方法 `__len__( )` 函数，如果有 `__len__( )` 则按 `__len__( )` 的值进行处理，如果该值为0则返回 `False`，否则 `True`，如果没有 `__len__( )`，则返回值为 `True`
- ☐ **C.** `bool( )` 函数如果没有参数，返回值为 `True`
- ☐ **D.** 表达式 `bool(int)` 的值为 `False`

2 判断题（每题 2 分，共 20 分）

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

第 1 题 小杨想写一个程序来算出正整数N有多少个因数，经过思考他写出了一个重复没有超过N/2次的循环就能够算出来了。（ ）

第 2 题 同样的整数序列分别保存在单链表和双向链中，这两种链表上的简单冒泡排序的复杂度相同。（ ）

第 3 题 在面向对象中，方法（Event）在Python的class中表现为class内定义的函数。( )

第 4 题 在下面的Python代码被执行将报错，因为newClass没有 `__init__( )` 魔术方法。（ ）

```
class newClass:
    pass

myNewClass = newClass()
```

第 5 题 如果某个Python对象（object）支持下标运算符（方括号运算符），则该对象在所对应class中定义了名为 `__getitem__` 的魔术方法。( )

第 6 题 深度优先搜索（DFS，Depth First Search的简写）属于图算法，其过程是对每一个可能的分支路径深入到不能再深入为止，而且每个节点只能访问一次。( )

第 7 题 哈夫曼编码（Huffman Coding）具有唯一性，因此有确定的压缩率。( )

第 8 题 Python虽然不支持指针和引用语法，但变量的本质是数据的引用（reference），因此可以实现各种C/C++数据结构。在下面Python代码中，由于删除了变量a，因此a所对应的数据也随之删除，故第4行代码被执行时，将报错。（ ）

```
a, b = [1,2,3], [3,4,5]
c = [a, b]
del a
print(c)
```

第 9 题 二叉搜索树查找的平均时间复杂度为$O(\log N)$。（ ）

第 10 题 二叉搜索树可以是空树（没有任何节点）或者单节点树（只有一个节点），或者多节点。如果是多节点，则左节点的值小于父节点的值，右节点的值大于父节点的值，由此推理，右节点树的值都大于根节点的值，左节点树的值都小于根节点的值。（ ）

3　编程题（每题 25 分，共 50 分）

3.1　编程题 1

- **试题名称**：闯关游戏
- **时间限制**：2.0 s
- **内存限制**：128.0 MB

3.1.1　问题描述

你来到了一个闯关游戏。

这个游戏总共有 N 关，每关都有 M 个通道，你需要选择一个通道并通往后续关卡。其中，第 i 个通道可以让你前进 a_i 关，也就是说，如果你现在在第 x 关，那么选择第 i 个通道后，你将直接来到第 $x + a_i$ 关（特别地，如果 $x + a_i \geq N$，那么你就通关了）。此外，当你顺利离开第 s 关时，你还将获得 b_s 分。

游戏开始时，你在第 0 关。请问，你通关时最多能获得多少总分？

3.1.2　输入描述

第一行两个整数 N, M，分别表示关卡数量和每关的通道数量。

接下来一行 M 个用单个空格隔开的整数 $a_0, a_1, \ldots, a_{M-1}$。保证 $1 \leq a_i \leq N$。

接下来一行 N 个用单个空格隔开的整数 $b_0, b_1, \ldots, b_{N-1}$。保证 $|b_i| \leq 10^5$。

3.1.3　输出描述

一行一个整数，表示你通关时最多能够获得的分数。

3.1.4　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。

3.1.5　样例输入 1

```
6 2
2 3
1 0 30 100 30 30
```

3.1.6　样例输出 1

```
131
```

3.1.7　样例解释 1

你可以在第 0 关选择第 1 个通道，获得 1 分并来到第 3 关；随后再选择第 0 个通道，获得 100 分并来到第 5 关；最后任选一个通道，都可以获得 30 分并通关。如此，总得分为 $1 + 100 + 30 = 131$。

3.1.8 样例输入 2

```
6 2
2 3
1 0 30 100 30 -1
```

3.1.9 样例输出 2

```
101
```

3.1.10 样例解释 2

请注意，一些关卡的得分可能是负数。

3.1.11 数据规模

对于 20% 的测试点，保证 $M = 1$。

对于 40% 的测试点，保证 $N \leq 20$；保证 $M \leq 2$。

对于所有测试点，保证 $N \leq 10^4$；保证 $M \leq 100$。

3.1.12 参考程序

```python
n, m = input().split()
n, m = int(n), int(m)

a = list(map(int, input().split()))
b = list(map(int, input().split()))

max_b = max(a)

f = [b[0]]

ans = - 10 ** 18

for i in range(1, n):
    f.append(None)
    for step in a:
        if i - step >= 0 and f[i - step] is not None:
            f[i] = max(f[i], f[i - step]) if f[i] is not None else f[i - step]
    if f[i] is not None:
        f[i] += b[i]
    if f[i] is not None and i + max_b >= n:
        ans = max(ans, f[i])
print(ans)
```

3.2 编程题 2

- **试题名称**：工作沟通
- **时间限制**：2.0 s
- **内存限制**：128.0 MB

3.2.1 问题描述

某公司有 N 名员工，编号从 0 至 $N-1$。其中，除了 0 号员工是老板，其余每名员工都有一个直接领导。我们假设编号为 i 的员工的直接领导是 f_i。

该公司有严格的管理制度，每位员工只能受到本人或本人直接领导或间接领导的管理。具体来说，规定员工 x 可以管理员工 y，当且仅当 $x=y$，或 $x=f_y$，或 x 可以管理 f_y。特别地，0 号员工老板只能自我管理，无法由其他任何员工管理。

现在，有一些同事要开展合作，他们希望找到一位同事来主持这场合作，这位同事必须能够管理参与合作的所有同事。如果有多名满足这一条件的员工，他们希望找到编号最大的员工。你能帮帮他们吗?

3.2.2 输入描述

第一行一个整数 N，表示员工的数量。

第二行 $N-1$ 个用空格隔开的正整数，依次为 $f_1, f_2, \ldots, f_{N-1}$。

第三行一个整数 Q，表示共有 Q 场合作需要安排。

接下来 Q 行，每行描述一场合作：开头是一个整数 m（$2 \le m \le N$），表示参与本次合作的员工数量；接着是 m 个整数，依次表示参与本次合作的员工编号（保证编号合法且不重复）。

保证公司结构合法，即不存在任意一名员工，其本人是自己的直接或间接领导。

3.2.3 输出描述

输出 Q 行，每行一个整数，依次为每场合作的主持人选。

3.2.4 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。

3.2.5 样例输入 1

```
5
0 0 2 2
3
2 3 4
3 2 3 4
2 1 4
```

3.2.6 样例输出 1

```
2
2
0
```

3.2.7 样例解释 1

对于第一场合作，员工 3,4 有共同领导 2，可以主持合作。

对于第二场合作，员工 2 本人即可以管理所有参与者。

对于第三场合作，只有 0 号老板才能管理所有员工。

3.2.8 样例输入 2

```
7
0 1 0 2 1 2
5
2 4 6
2 4 5
3 4 5 6
4 2 4 5 6
2 3 4
```

3.2.9 样例输出 2

```
2
1
1
1
0
```

3.2.10 数据规模

对于 50% 的测试点，保证 $N \leq 50$。

对于所有测试点，保证 $3 \leq N \leq 300$；$Q \leq 100$。

3.2.11 参考程序

```python
n = int(input())
father = [-1] + list(map(int, input().split(' ')))
child = [[] for i in range(n)]
for i in range(1, n):
    child[father[i]].append(i)

q = int(input())
for _ in range(q):
    x = list(map(int, input().split(' ')))[1:]
    ans = -1
    for r in range(n):
        vis = [False for i in range(n)]
        def dfs(node):
            vis[node] = True
            for c in child[node]:
                dfs(c)
        dfs(r)
        flag = True
        for y in x:
            if not vis[y]:
                flag = False
                break
        if flag:
            ans = r
    print(ans)
```