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

Python 六级

2026 年 09 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	A	C	B	A	D	C	B	B	A	B	C	A

第 1 题 下列代码执行后的输出结果是（ ）。

```
class Animal:
    def speak(self):
        print("Animal ", end="")

class Cat(Animal):
    def speak(self):
        print("Cat ", end="")

def main():
    p: Animal = Cat()
    p.speak()

main()
```

- A. `Animal`
- B. `Cat`
- C. `Animal Cat`
- D. 编译错误

第 2 题 下列代码横线处填写哪一项，才可以正确调用父类的带参数构造函数（ ）

```
class Machine:
    def __init__(self, s):
        self.id = s

class Robot(Machine):
    def __init__(self, s, n):
        ---------------
        self.level = n
```

- A. `Machine(s)`
- B. `Machine.Machine(s)`
- C. `super().__init__(s)`
- D. `self.id = s`

第 3 题 下列代码执行后的输出顺序是（ ）。

```python
class Base:
    def __init__(self):
        print("B ", end="")
    def __del__(self):
        print("~B ", end="")

class Derived(Base):
    def __init__(self):
        super().__init__()
        print("D ", end="")
    def __del__(self):
        print("~D ", end="")

def main():
    p = Derived()
    del p

main()
```

- [ ] **A.** `B D ~D ~B`
- [ ] **B.** `B D ~D`
- [ ] **C.** `D B ~D ~B`
- [ ] **D.** `B D`

第 4 题 下列代码执行后的输出结果是（　　）。

```python
s = []
q = []

for i in range(2,7,2):
    s.append(i)
    q.append(i)

s.pop()
q.pop(0)

print(s[-1], q[0])
```

- [ ] **A.** `4 4`
- [ ] **B.** `2 2`
- [ ] **C.** `6 6`
- [ ] **D.** `6 2`

第 5 题 下面循环队列采用牺牲一个存储单元（空出一个位置）的方法区分队空与队满，`full()` 函数判断队列是否已满，横线处应填写（　　）。

```python
MAXN = 8
data = [0] * MAXN
front = 0
rear = 0

def full():
    return _____________
```

- [ ] **A.** `rear == front`
- [ ] **B.** `(front + 1) % MAXN == rear`

- [ ] **C.** `(rear + 1) % MAXN == front`
- [ ] **D.** `rear == MAXN - 1`

第 6 题 下列函数属于二叉树哪一种遍历（　　）

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

def visit(root):
    if root is None:
        return
    visit(root.left)
    print(root.val, end=" ")
    visit(root.right)
```

- [ ] **A.** 前序遍历
- [ ] **B.** 中序遍历
- [ ] **C.** 后序遍历
- [ ] **D.** 层序遍历

第 7 题 已知一棵二叉树的先序遍历序列为 `A B D E C F`，中序遍历序列为 `D B E A C F`，则其后序遍历序列是（　　）。

- [ ] **A.** `D E B F C A`
- [ ] **B.** `D B E F C A`
- [ ] **C.** `E D B F C A`
- [ ] **D.** `D E B C F A`

第 8 题 下面函数用来计算二叉树的高度，横线处应填写（　　）。

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

def height(root):
    if root is None:
        return 0
    leftH = height(root.left)
    rightH = height(root.right)
    return _______________
```

- [ ] **A.** `leftH + rightH`
- [ ] **B.** `min(leftH, rightH) + 1`
- [ ] **C.** `max(leftH, rightH)`
- [ ] **D.** `max(leftH, rightH) + 1`

第 9 题 下面代码实现二叉树深度优先搜索（迭代版前序遍历），横线处应当填写（　　）。

```python
class TreeNode:
    def __init__(self, val):
        self.value = val
        self.left = None
        self.right = None

def dfs(root):
    if root is None:
        return
    s = []
    s.append(root)
    while len(s) > 0:
        node = s.pop()
        print(node.value, end=" ")
        # --------此处填空--------
```

- [ ] **A.**

```python
s.append(node.right)
s.append(node.left)
```

- [ ] **B.**

```python
s.append(node.left)
s.append(node.right)
```

- [ ] **C.**

```python
if node.right:
    s.append(node.right)
if node.left:
    s.append(node.left)
```

- [ ] **D.**

```python
if node.left:
    s.append(node.left)
if node.right:
    s.append(node.right)
```

第 10 题 下面函数用于在二叉搜索树里面查找数值 `x`，横线处应当填写（　　）。

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

def searchBST(root, x):
    if root is None or root.val == x:
        return root
    if x < root.val:
        return searchBST(root.left, x)
    return _____________
```

- [ ] **A.** `searchBST(root.left, x)`
- [ ] **B.** `searchBST(root.right, x)`
- [ ] **C.** `searchBST(root, x + 1)`
- [ ] **D.** `root.right`

第 11 题 有 5 个字符，其出现频率分别为 `{2, 4, 5, 9, 12}`。按哈夫曼算法构造编码树，其最小带权路径长度 WPL 为（　　）。

- [ ] **A.** 68
- [ ] **B.** 69
- [ ] **C.** 71
- [ ] **D.** 73

第 12 题 下面代码使用反射法生成 `n` 位格雷编码，横线处应该填写（　　）

```
def gray(n):
    ans = ["0", "1"]
    for bit in range(2, n + 1):
        oldSize = len(ans)
        for i in range(oldSize - 1, -1, -1):
            ans.append(_________________)
        for i in range(oldSize):
            ans[i] = "0" + ans[i]
    return ans
```

- [ ] **A.** `"1" + ans[i]`
- [ ] **B.** `ans[i]`
- [ ] **C.** `"1" + str(i)`
- [ ] **D.** `"0" + ans[i]`

第 13 题 下面代码计算爬到第 `n` 级台阶的方案数，每次可以跨 1 级或者 2 级台阶，横线处应填写（　　）。

```
def ways(n):
    if n <= 2:
        return n
    dp = [0] * (n + 1)
    dp[1] = 1
    dp[2] = 2
    for i in range(3, n + 1):
        dp[i] = ________________
    return dp[n]
```

- [ ] **A.** `dp[i - 1] + 1`
- [ ] **B.** `dp[i - 1] + dp[i - 2]`
- [ ] **C.** `dp[i - 2] + 2`
- [ ] **D.** `2 * dp[i - 1]`

第 14 题 代码求解：从包含非负元素的数组选取若干**互不相邻**的元素，得到最大和，横线处填写（　　）

```python
def maxSum(a):
    n = len(a)
    if n == 0:
        return 0
    if n == 1:
        return a[0]
    dp = [0] * n
    dp[0] = a[0]
    dp[1] = max(a[0], a[1])
    for i in range(2, n):
        dp[i] = _______________
    return dp[-1]
```

- A. `dp[i - 1] + a[i]`
- B. `dp[i - 2] + a[i]`
- C. `max(dp[i - 1], dp[i - 2] + a[i])`
- D. `max(dp[i - 1], a[i])`

第 15 题 下面是一维数组实现的 0/1 背包，内层循环必须**从大到小枚举容量**，主要原因是（　　）

```python
# n物品数量，W背包总容量
for i in range(n):
    w = W
    while w >= weight[i]:
        dp[w] = max(dp[w], dp[w - weight[i]] + value[i])
        w -= 1
```

- A. 保证每件物品最多被选择一次
- B. 保证物品必须按照重量从大到小选择
- C. 降低时间复杂度到 $O(n)$
- D. 防止数组 `dp` 发生越界

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 下面这段描述：代码可以运行时出错，编译器会自动为 `Student` 生成无参构造函数。

```python
class Student:
    def __init__(self, x):
        self.age = x

def main():
    s = Student()  # 尝试调用无参构造

main()
```

第 2 题 下面代码是合法的，派生类可以直接访问基类的私有成员 `value` 。

```python
class Base:
    def __init__(self):
        self.__value = 10

class Child(Base):
    def get(self):
        return self.__value

c = Child()
c.get()
```

第 3 题 下列代码执行后，输出结果为 `30` 。

```python
from collections import deque
q = deque()
q.append(10)
q.append(20)
q.append(30)
q.popleft()
print(q[0])
```

第 4 题 一棵完全二叉树按照从上到下、从左到右的顺序，将节点依次存储在数组 `tree[1]` 、 `tree[2]` 、…… 中。若节点 `tree[i]` 存在左孩子，则其左孩子存储在 `tree[2*i]` 中。

第 5 题 对任意一棵左节点更小的二叉搜索树执行中序遍历，得到的关键字序列一定是非递减的。

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

def inorder(root):
    if not root:
        return
    inorder(root.left)
    print(root.val, end=" ")
    inorder(root.right)
```

第 6 题 从结点 `start` 出发运行下面代码访问一棵树；第一次抵达某个结点时经过的边数，一定是 `start` 到达该结点的最少边数。

```python
from collections import deque

# tree为邻接表
def search(start, tree):
    n = 100
    visited = [False] * n
    dist = [0] * n
    q = deque()
    q.append(start)
    visited[start] = True
    dist[start] = 0

    while q:
        u = q.popleft()
        for v in tree[u]:
            if not visited[v]:
                visited[v] = True
                dist[v] = dist[u] + 1
                q.append(v)
```

第 7 题 哈夫曼编码的生成过程基于贪心算法，出现频率越高的字符，其编码长度一定不会比出现频率更低的字符更长。

第 8 题 在 `n` 位格雷码中，任意两个编码之间都只相差一个二进制位。

第 9 题 下列一维 DP 代码实现的是完全背包问题，处理第 `i` 种物品时，同一件物品有可能被重复选取。

```python
for i in range(n):
    w = weight[i]
    while w <= W:
        dp[w] = max(dp[w], dp[w - weight[i]] + value[i])
        w += 1
```

第 10 题 下面递归函数可以算出正确的斐波那契数值，该代码时间复杂度与空间复杂度均为 $O(n)$。

```python
def fib(n):
    if n <= 1:
        return n
    return fib(n - 1) + fib(n - 2)
```

3　编程题（每题 25 分，共 50 分）

3.1　编程题 1

- **试题名称**：数组划分
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1　题目描述

给定 n 个整数构成的数组 $A = [a_1, a_2, \ldots, a_n]$。

你需要将数组 A 划分为若干非空连续子段。对于划分得到的某个子段，它的偏差值定义为子段内整数和的平方。划分方案的偏差值定义为所有子段偏差值之和。

你需要最小化划分方案的偏差值。

形式化地，你可以将 A 划分为若干非空连续子段 $A_1, A_2, \ldots, A_k$，使得 $A = A_1 + A_2 + \ldots A_k$，这里的 $+$ 代表数组的连接。对于 $1 \le i \le k$，设数组 $A_i = [a_1^{(i)}, \ldots, a_{m_i}^{(i)}]$ 包含 m_i 个整数。你需要最小化 $\sum_{i=1}^{k} \left(\sum_{j=1}^{m_i} a_j^{(i)} \right)^2$。

3.1.2 输入格式

第一行，一个正整数 n，表示数组 A 的长度。

第二行，n 个整数 $a_1, a_2, \ldots, a_n$，表示数组 A。

3.1.3 输出格式

一行，一个整数，表示划分方案偏差值的最小值。

3.1.4 样例

3.1.5 输入样例 1

```
4
1 2 -3 4
```

3.1.6 输出样例 1

```
6
```

3.1.7 输入样例 2

```
6
-1 -1 4 -5 -1 4
```

3.1.8 输出样例 2

```
0
```

3.1.9 数据范围

对于 40% 的测试点，保证 $0 \le a_i \le 50$。

对于所有测试点，保证 $1 \le n \le 2000$，$-100 \le a_i \le 100$。

3.1.10 参考程序

```python
n = int(input())

a = [0] + list(map(int, input().split()))
# 提前准备好，从a_1~a_i之和是多少
pre = [sum(a[:i + 1]) for i in range(n + 1)]
# f[i]: 从a_1到a_i最小偏差值
f = [0]
for i in range(1, n + 1):
    # 假设最后一个分段是a_{j+1}~a_i，那么偏差值多少，从这些值里面选最小值
    f.append(min(f[j] + (pre[i] - pre[j]) ** 2 for j in range(i)))
print(f[n])
```

3.2 编程题 2

- **试题名称**：分树规划
- **时间限制**：1.0 s

- **内存限制**：512.0 MB

3.2.1 题目描述

老师有一棵有 n 个结点的树，结点依次以 $1, 2, \ldots, n$ 编号。

老师想将这棵树作为奖品分给两位同学。具体而言，老师会选择一条边并从树上删去它，从而将这棵树分为两个连通块。两位同学分别可以得到其中一个连通块。

如果有同学拿到的连通块结点数明显小于另一位同学，那么这位同学会不太高兴。为了避免这种情况出现，老师想知道两个连通块结点数之差的绝对值最小是多少。

3.2.2 输入格式

第一行，一个正整数 n，表示结点数量。

接下来 $n-1$ 行，每行两个正整数 u_i, v_i，表示一条连接结点 u_i 和结点 v_i 的边。

3.2.3 输出格式

输出一行，一个整数，表示答案。

3.2.4 样例

3.2.5 输入样例 1

```
4
1 2
2 3
3 4
```

3.2.6 输出样例 1

```
0
```

3.2.7 输入样例 2

```
6
1 2
1 3
1 4
1 5
5 6
```

3.2.8 输出样例 2

```
2
```

3.2.9 数据范围

对于 40% 的测试点，保证 $2 \leq n \leq 500$。

对于所有测试点，保证 $2 \leq n \leq 2 \times 10^4$。

3.2.10 参考程序

```python
n = int(input())
e = [[] for _ in range(n + 1)]
for i in range(1, n):
    u, v = map(int, input().split())
    e[u].append(v)
    e[v].append(u)
# 初始化答案数组，默认每一个节点被断开初始答案都是n
ans = [n] * (n + 1)
def dfs(u, p=0):
    # 计算以节点u为根节点的子树的总结点数
    # 等于各个子节点为根的子树的总结点数之和
    sz = 1 + sum(dfs(i, u) for i in e[u] if i != p)
    # 假设我们把连接节点u和根节点的边断开，那么我们的节点数之差的绝对值就是这么多
    ans[u] = abs(n - 2 * sz)
    return sz
dfs(1)
print(min(ans))
```