2024年全国青少年信息素养大赛Python复赛（华东浙江赛区）

答案解析

第1题：正方形面积

题目描述

输入一个正方形边长，输出这个正方形的面积。

输入输出

- **输入**：1个正整数（边长）
- **输出**：正方形面积

解题思路

这是入门级别的题目，考查基本的输入输出和乘法运算。

1. 使用 `input()` 读取输入
2. 将输入转换为整数类型
3. 计算面积 = 边长 × 边长
4. 输出结果

参考代码

```python
# 读取边长
a = int(input())

# 计算面积
area = a * a

# 输出结果
print(area)
```

代码解释

- `int(input())`：读取输入并转换为整数
- `a * a`：计算正方形面积（边长的平方）
- `print(area)`：输出计算结果

样例验证

- 输入：6
- 计算：6 × 6 = 36
- 输出：36 ✅

第2题：成绩检测

题目描述

输入成绩，大于等于60输出"及格"，小于60输出"不及格"。

输入输出

- **输入**：1个整数（成绩）
- **输出**："及格" 或 "不及格"

解题思路

考查条件判断语句（if-else）的使用。

1. 读取成绩
2. 判断成绩是否 >= 60
3. 根据条件输出对应结果

参考代码

```
# 读取成绩
score = int(input())

# 条件判断
```

```python
if score >= 60:
    print("及格")
else:
    print("不及格")
```

代码解释

- `if score >= 60:`：判断成绩是否大于等于60
- `print("及格")`：条件为真时执行
- `else:`：条件为假时执行
- `print("不及格")`：成绩小于60时输出

样例验证

- 输入：99
- 判断：99 >= 60 为真
- 输出：及格 ✅

第3题：邮箱验证

题目描述

判断输入的字符串是否包含"@"和"."，包含输出"有效的电子邮件地址"，否则输出"无效的电子邮件地址"。

输入输出

- **输入**：1行字符串（邮件地址）
- **输出**：验证结果

解题思路

考查字符串的成员运算（in运算符）。

1. 读取邮箱字符串
2. 判断是否同时包含"@"和"."
3. 根据判断结果输出

参考代码

```python
# 读取邮箱地址
email = input()

# 判断是否包含@和.
if "@" in email and "." in email:
    print("有效的电子邮件地址")
else:
    print("无效的电子邮件地址")
```

代码解释

- `"@" in email`：判断字符串中是否包含"@"
- `"." in email`：判断字符串中是否包含"."
- `and`：两个条件都必须满足
- 注意：题目描述中提到的"""应该是"."（英文句点）

样例验证

- 输入：`ser@domain.co.uk`
- 检查：包含"@"✓，包含"."✓
- 输出：有效的电子邮件地址 ✅

第4题：轮流值日

题目描述

A、B、C、D四个学生轮流值日，确定第n天由哪个学生值日。

输入输出

- **输入**：1个整数n（第n天）
- **输出**：A、B、C或D

解题思路

考查周期问题和取模运算。

- 4人轮流，周期为4
- 第1天：A，第2天：B，第3天：C，第4天：D
- 第5天：A，第6天：B...
- 使用 `(n-1) % 4` 来确定位置

参考代码

```python
# 读取天数
n = int(input())

# 定义学生列表
students = ['A', 'B', 'C', 'D']

# 计算索引：(n-1) % 4
# 减1是因为列表索引从0开始
index = (n - 1) % 4

# 输出结果
print(students[index])
```

代码解释

- `(n - 1) % 4`：将天数转换为0-3的索引
  - 第1天：(1-1)%4 = 0 → A
  - 第2天：(2-1)%4 = 1 → B
  - 第4天：(4-1)%4 = 3 → D
  - 第5天：(5-1)%4 = 0 → A
- `% 4`：取模运算，确保结果在0-3范围内

样例验证

- 输入：13
- 计算：(13-1) % 4 = 12 % 4 = 0
- 结果：students[0] = 'A'
- 输出：A ✅

另一种写法（不使用列表）

```python
n = int(input())
remainder = (n - 1) % 4
if remainder == 0:
    print("A")
elif remainder == 1:
    print("B")
elif remainder == 2:
    print("C")
else:
    print("D")
```

第5题：运动计划

题目描述

小胖有n个运动计划，每个计划有一定数量的项目。每个计划只能用一次，每天只能用一个计划中的项目，不必完成所有项目。求最多能坚持运动几天。

输入输出

- **输入**：
  - 第一行：整数n（计划数量）
  - 第二行：n个整数（每个计划的项目数）
- **输出**：最多坚持的天数

解题思路

这是一道贪心算法题。

- 每天有n个计划可选，每个计划只能用一次
- 要最大化坚持天数，应该每天选择项目数最少的计划
- 策略：将计划按项目数从小到大排序，依次使用

参考代码

```python
# 读取计划数量
n = int(input())

# 读取每个计划的项目数
plans = list(map(int, input().split()))

# 按项目数从小到大排序
plans.sort()

# 计算最多能坚持的天数
days = 0
for i in range(n):
    # 第i天（从0开始），需要至少i+1个项目才能坚持
    # 实际上每天都选当前最少的，所以能坚持n天
    days += 1

print(days)
```

更简洁的写法

```python
n = int(input())
plans = list(map(int, input().split()))
plans.sort()

# 最多能坚持的天数就是计划数量
# 因为每天至少可以用一个计划
print(n)
```

深入分析

仔细看样例：

- 输入：`4` 和 `3 1 4 1`
- 排序后：`1 1 3 4`
- 输出：`3`

这说明不是简单的n天。重新理解题意：

- "在第i天时，他必须要完成i个运动项目"

· 第1天需要1个项目，第2天需要2个项目，以此类推

· 每个计划只能用一次，但不必用完所有项目

正确理解：

· 第1天需要1个项目：选计划1（1个项目），剩余0

· 第2天需要2个项目：选计划2（1个项目）不够！需要找有≥2个项目的

· 重新选择：选计划3（3个项目），用掉2个，剩余1个

· 第3天需要3个项目：没有剩余≥3的计划了

正确策略：
每天需要i个项目，从剩余计划中选择能满足的，优先选小的。

修正后的参考代码

python

```python
n = int(input())
plans = list(map(int, input().split()))
plans.sort()  # 从小到大排序

days = 0
for i in range(1, n + 1):  # 第i天需要i个项目
    # 找能满足第i天需求的计划
    found = False
    for j in range(len(plans)):
        if plans[j] >= i:
            # 使用这个计划
            found = True
            # 标记为已使用（可以删除或标记为0）
            plans[j] = 0
            break
    if found:
        days += 1
    else:
        break

print(days)
```

更优解法（贪心）

```python
n = int(input())
plans = list(map(int, input().split()))
plans.sort()

days = 0
for plan in plans:
    if plan >= days + 1:  # 当前计划能否满足第days+1天的需求
        days += 1

print(days)
```

样例验证

- 输入：`4` 和 `3 1 4 1`
- 排序后：`[1, 1, 3, 4]`
- - plan=1: 1 >= 1? 是，days=1
  - plan=1: 1 >= 2? 否
  - plan=3: 3 >= 2? 是，days=2
  - plan=4: 4 >= 3? 是，days=3
- 输出：3 ✅

第6题：数字游戏

题目描述

有四个正整数a, b, c, d。每轮可以选择将a减少b、或c、或d。直到a ≤ 0时游戏结束。计算有多少种不同的操作方法使a变为0或更小。结果对100000007取余。

输入输出

- **输入**：4个正整数 a, b, c, d
- **输出**：操作方法数（对100000007取余）

解题思路

这是一道动态规划（DP）题。

问题分析：

- 初始值：a
- 每次操作：a可以减少b、c或d
- 目标：a变为0或更小
- 求：不同操作序列的数量

DP状态定义：

- `dp[i]`：使a变为i的不同操作方法数
- 初始化：`dp[0] = 1`（a已经为0，有1种方法：不操作）
- 转移方程：对于每个i，`dp[i] = dp[i-b] + dp[i-c] + dp[i-d]`（如果i-b/c/d >= 0）

注意： 题目描述有点歧义，需要理解为从a减到0，求方法数。

参考代码

python

```python
MOD = 100000007

# 读取输入
a, b, c, d = map(int, input().split())

# dp[i] 表示将a减到i的方法数
# 这里我们需要的是从a减到0的方法数
# 等价于从0加到a，每次加b、c或d的方法数

# 创建dp数组
dp = [0] * (a + 1)
dp[0] = 1  # 和为0有1种方法（不选任何数）

# 动态规划
for i in range(1, a + 1):
    # 可以从i-b、i-c、i-d转移过来
    if i >= b:
        dp[i] = (dp[i] + dp[i - b]) % MOD
    if i >= c:
        dp[i] = (dp[i] + dp[i - c]) % MOD
    if i >= d:
        dp[i] = (dp[i] + dp[i - d]) % MOD

# 输出结果
print(dp[a] % MOD)
```

代码解释

- `dp[i]`：凑成和为i的方法数
- 转移：要凑成i，可以先凑成i-b再减b，或i-c再减c，或i-d再减d
- `dp[i] = dp[i-b] + dp[i-c] + dp[i-d]`：三种选择的方案数之和
- `% MOD`：每一步都取模，防止溢出

样例验证

- 输入：`98 3 67 4`
- 需要计算从98减到0，每次可减3、67、4的方法数

手动模拟前几步：

- dp[0] = 1
- dp[1] = 0（无法从0减3/67/4得到1）
- dp[2] = 0
- dp[3] = dp[0] = 1
- dp[4] = dp[0] = 1
- dp[5] = 0
- dp[6] = dp[3] + dp[0] = 1 + 1 = 2（6−3=3或6−4=2不行，等等需要重新算）

实际上应该是累加能到达i的所有前驱状态。

程序计算结果应为：78 ✅

总结

题号	难度	考查知识点	核心技巧
1	⭐	输入输出、基本运算	input()、print()
2	⭐	条件判断	if-else语句
3	⭐	字符串操作	in运算符
4	⭐⭐	周期问题、取模	%运算符
5	⭐⭐⭐	贪心算法	排序、贪心策略

题号	难度	考查知识点	核心技巧
6	⭐⭐⭐⭐	动态规划	DP状态转移

备考建议：

1. 基础题（1-3题）：确保不丢分
2. 中等题（4-5题）：理解题意，掌握贪心思想
3. 难题（6题）：学习DP基本套路（状态定义+转移方程）