全国青少年信息素养大赛-2026-初中组-算法-Py-数字-复赛-模拟题-01-答案与解析

1	2	3	4	5	6	7	8	9	10
B	A	A	B	B	A、B、C、D	A、B、D	A、B、C、D	A、B、C、D	A、B、D

一、单选题

（1）文物数字化中心在复制非遗传承人档案（嵌套列表）时使用了 `copy()` 方法。执行以下代码后，输出结果为（ ）。

```
a = [1, 2, [3, 4]]
b = a.copy()
b[2][0] = 99
print(a[2][0])
```

A. `3`

B. `99`

C. `[3, 4]`

D. `0`

【正确答案】B

【题目解析】

✅ 正确答案

B（`99`）

💡 核心考点

浅拷贝 `copy()` 不复制嵌套对象。

🔎 执行追踪

`b` 是 `a` 的浅拷贝，内层列表 `[3,4]` 仍与 `a` 共享引用，修改 `b[2][0]` 同步影响 `a[2][0]`。

⚠️ 易错提醒

浅拷贝只复制最外层，深拷贝需使用 `copy.deepcopy()` 。

（2）非遗保护中心比对两批数字化文物编号，需要找出两批共有的编号（交集）和全部出现过的编号（并集）。执行以下代码后，输出结果为（ ）。

```
a = {1, 2, 3}
b = {2, 3, 4}
print(a & b)
print(a | b)
```

A. `{2, 3}` 和 `{1, 2, 3, 4}`
B. `{1, 4}` 和 `{1, 2, 3, 4}`
C. `{2, 3}` 和 `{1, 2, 3, 4, 2, 3}`
D. `{1, 2, 3, 4}` 和 `{2, 3}`

【正确答案】A

【题目解析】

✅ 正确答案

A

💡 核心考点

集合的交集 `&` 与并集 `|` 运算。

🔎 执行追踪

`a & b` 取共同元素得 `{2, 3}`；`a | b` 合并去重得 `{1, 2, 3, 4}` 。

⚠️ 易错提醒

集合自动去重，并集不会重复出现元素。

（3）数字博物馆按藏品数字化处理公式计算各展厅工作量，支持通过关键字指定参数。执行以下代码后，输出结果为（ ）。

```
def calc(x, y=2):
    return x ** y

print(calc(y=3, x=2))
```

A. `8`
B. `6`
C. `9`
D. 程序会报错

【正确答案】A

【题目解析】

✅ 正确答案

A（`8`）

💡 核心考点

关键字参数传递。

🔎 执行追踪

`x=2, y=3`，计算 $2^3 = 8$ 。

⚠️ 易错提醒

关键字参数可以打乱顺序传入，不影响赋值。

（4）文化遗产保护中心统计了各项目修复评分，需要按成绩降序排列找出排名第一的项目。执行以下代码后，输出结果为（ ）。

```
scores = [('皮影戏', 90), ('剪纸', 85), ('苏绣', 95)]
scores.sort(key=lambda x: x[1], reverse=True)
print(scores[0])
```

A. `('皮影戏', 90)`
B. `('苏绣', 95)`
C. `('剪纸', 85)`
D. `95`

【正确答案】B

【题目解析】

✅ 正确答案

B `('苏绣', 95)`

💡 核心考点

考察 sort() 配合 lambda 按元组指定元素降序排序。

🔎 执行追踪

按每个元组第 2 个元素降序排序后，列表变为 `[('苏绣', 95), ('皮影戏', 90), ('剪纸', 85)]`，取首元素。

⚠️ 易错提醒

`reverse=True` 表示降序；输出的是整个元组而非分数。

（5）数字化保护中心用函数同时返回文物统计中的最小值和最大值。执行以下代码后，输出结果为（ ）。

```
def min_max(nums):
    return min(nums), max(nums)

result = min_max([5, 2, 8, 1])
print(type(result))
print(result[0])
```

A. `<class 'list'>` 和 `1`

B. `<class 'tuple'>` 和 `1`

C. `<class 'tuple'>` 和 `5`

D. `<class 'list'>` 和 `5`

【正确答案】B

【题目解析】

✅ 正确答案

B

💡 核心考点

函数返回多个值时默认打包为元组。

🔎 执行追踪

`min_max` 返回 `(1, 8)`，类型为 `tuple`，`result[0]` 为 `1`。

⚠️ 易错提醒

用逗号分隔的多返回值是元组（tuple），不是列表（list）。

二、多选题

（6）文物数字化中心在复制非遗保护档案时，需要理解深浅拷贝的区别。关于 Python 的深浅拷贝，以下说法正确的有（ ）。

A. `copy()` 是浅拷贝，只复制最外层对象
B. 浅拷贝后的嵌套列表与原列表共享内部的子列表
C. 使用 `deepcopy()` 可以创建完全独立的副本（需要 `import copy`）
D. 对不可变对象（如整数）组成的列表，浅拷贝和深拷贝效果相同

【正确答案】A、B、C、D

【题目解析】

✅ 正确答案

A、B、C、D

💡 核心考点

浅拷贝与深拷贝的区别。

🔎 执行追踪

`copy()` 仅复制最外层，嵌套子列表仍为引用共享；`deepcopy()` 递归复制所有层级，需 `import copy`；不可变对象不存在修改问题，两者效果等同。

⚠️ 易错提醒

浅拷贝对嵌套结构修改子列表会影响原列表，这是最常见的坑。

（7）数字加密组用递归算法生成文物数字档案的密码排列。关于递归函数，以下说法正确的有（ ）。

A. 递归函数必须至少有一个终止条件（基准情况）
B. 没有终止条件的递归会导致栈溢出
C. 递归函数不能有返回值
D. 任何递归问题都可以用循环来等价实现

【正确答案】A、B、D

【题目解析】

✅ 正确答案

A、B、D

💡 核心考点

递归函数的基本性质与特征。

🔎 执行追踪

A 正确，递归必须有基准情况以终止；B 正确，无终止条件会无限调用导致栈溢出；C 错误，递归函数

可以有返回值；D 正确，递归与迭代在理论上是等价的，任何递归都可改写为循环（可能需要借助栈）。

⚠️ 易错提醒

不要误以为递归函数不能返回值，实际上大多数递归函数都通过返回值传递结果。

（8）非遗保护中心在解密过程中使用完善的异常处理结构。关于 `try-except-else-finally` 异常处理结构，以下说法正确的有（ ）。

A. `else` 子句只在 `try` 块没有发生异常时执行
B. `finally` 子句无论是否发生异常都会执行
C. `except` 可以有多个，用于捕获不同类型的异常
D. `else` 必须出现在所有 `except` 之后、`finally` 之前

【正确答案】A、B、C、D

【题目解析】

✅ 正确答案

A、B、C、D

💡 核心考点

考察 `try-except-else-finally` 异常处理结构的语法与执行顺序。

🔎 执行追踪

`try` 正常→执行 `else`；无论异常与否→执行 `finally`；多个 `except` 按顺序匹配；语法顺序固定为 `try→except→else→finally`。

⚠️ 易错提醒

`else` 只在无异常时触发，且必须位于所有 `except` 之后、`finally` 之前，顺序不可颠倒。

（9）文物数字化中心需要将非遗档案写入文件并妥善保管。关于文件操作，以下说法正确的有（ ）。

A. `'r'` 模式表示以只读方式打开文件
B. `'w'` 模式会覆盖文件原有内容
C. `'a'` 模式表示在文件末尾追加内容
D. 使用 `with` 语句打开文件，可以自动关闭文件

【正确答案】A、B、C、D

【题目解析】

✅ 正确答案

A、B、C、D

💡 核心考点

Python 文件打开模式与 `with` 语句的用法。

🔎 执行追踪

`'r'` 只读、`'w'` 写入并覆盖、`'a'` 末尾追加，`with` 语句会在代码块结束后自动调用 `close()` 关闭文件，四项描述均正确。

⚠️ 易错提醒

注意 `'w'` 模式会清空原文件内容，若需保留旧数据应使用 `'a'` 模式。

（10）数字修复组在文化遗产保护区域搜索最优修复路线。关于搜索算法，以下说法正确的有（ ）。

A. DFS（深度优先搜索）使用栈或递归实现

B. BFS（广度优先搜索）使用队列实现

C. DFS 一定能找到最短路径

D. 在网格搜索中，通常使用方向数组（如 `dx = [0,0,1,-1], dy = [1,-1,0,0]`）来遍历四个方向的相邻位置

【正确答案】A、B、D

【题目解析】

✅ 正确答案

A、B、D

*1. 🧠 解题思路

本题考查搜索算法（DFS 与 BFS）的基本特性及网格搜索的常用技巧。

*2. ⚙️ 逻辑推演

A 正确：DFS 沿一条路径深入到底再回溯，天然适合用递归实现，递归本质上利用了系统调用栈；也可以显式使用栈来模拟。

B 正确：BFS 按层扩展，需要先进先出的数据结构，使用队列（queue）实现。

C 错误：DFS 找到的路径不一定最短，它只保证能找到一条可行路径。求最短路径在无权图中通常使用 BFS。

D 正确：方向数组是网格搜索中的常用技巧，通过 `dx`、`dy` 数组配合循环即可简洁地遍历上下左右四个相邻格子，避免重复书写四段判断代码。

*3. ✅ 参考代码

```python
# 网格搜索中方向数组的典型用法
dx = [0, 0, 1, -1]
dy = [1, -1, 0, 0]
x, y = 2, 3  # 当前位置
for i in range(4):
    nx, ny = x + dx[i], y + dy[i]  # 计算四个相邻位置
    # 接下来判断（nx, ny）是否合法并继续搜索
```

*4. ⚠️ 警示与分析

易错点：

误以为 DFS 也能找最短路径。实际上 DFS 是"一条路走到黑"，找到的第一条路径未必最短；BFS 在无权图中才能保证最短。

混淆 DFS 与 BFS 的数据结构：DFS 用栈（或递归），BFS 用队列，切勿弄反。

三、编程题

（11）Y3694 数字文物库房

题目描述

数字文物库房负责管理珍贵文物的数字化资源存取。假设对于任意一类文物资源，每天开始工作时的存储容量已知，并且一天之内不会通过采购的方式增加。每天会有多位研究人员前来申请下载，每位研究人员希望下载不同数量的资源。如果申请人需要的数量超过了当时的库存量，库房就会拒绝该请求。

请你编写程序：输入资源初始库存、申请人人数及每位申请人所需的数量，计算有多少位申请人没有申请到资源。

输入描述

共 3 行：

- 第一行是每天开始时的文物资源总量 m ；
- 第二行是这一天申请的人数 n （ $0 < n \leq 100$ ）；
- 第三行共有 n 个数，分别记录了每位申请人希望取走的资源数量（按照先后顺序）。

输出描述

一行，一个整数，表示没有申请到资源的人数。

输入样例 1

```
30
6
10 5 20 6 7 8
```

输出样例 1

```
1   2
```

【正确答案】见解析

【题目解析】

✅ 解题思路

本题是经典的贪心模拟问题。按照申请顺序依次处理，维护剩余库存，当库存不足时计数拒绝人数。

💡 核心考点

贪心算法模拟、累加与判断。

🔎 执行追踪

依次处理每个申请：库存30→申请人1取10剩20→申请人2取5剩15→申请人3需20但剩15，拒绝（计数1）→申请人4取6剩9→申请人5取7剩2→申请人6需8但剩2，拒绝（计数2），共拒绝2人。

⚠️ 易错提醒

注意是"当前库存不足时才拒绝"，不是累计需求超过总量就拒绝。

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

rejected = 0
remaining = m

for req in requests:
    if req <= remaining:
        remaining -= req
    else:
        rejected += 1

print(rejected)
```

（12）Y3695 非遗密码拼图

题目描述

非遗数字保护中心使用火柴棒在展厅中摆出数字密码，向公众展示传统技艺的编号。用火柴棒拼数字 $0-9$ 的拼法为：0 需 6 根、1 需 2 根、2 需 5 根、3 需 5 根、4 需 4 根、5 需 5 根、6 需 6 根、7 需 3 根、8 需 7 根、9 需 6 根。

现在有 n 根火柴棒，非遗传承人要用它们拼出形如 `A+B=C` 的等式密码。注意：

1. 加号和等号各自需要 2 根火柴棒；

2. 如果 A 不等于 B，则 `A+B=C` 或 `B+A=C` 视为不同的等式（ A 、 B 、 $C \geq 0$ ）；

3. n 根火柴棒必须全部用上。

请你编写程序，计算能拼成的不同等式的数目。

输入描述

一行，有一个整数 n （ $1 \leq n \leq 24$ ）。

输出描述

一行，有一个整数，表示能拼成的不同等式的数目。

输入样例 1

```
14
```

输入样例 2

```
18
```

输出样例 1

```
2
```

输出样例 2

```
9
```

【正确答案】见解析

【题目解析】

✅ 解题思路

本题是枚举+哈希表的综合题。先预处理每个数字（0~11111范围）所需火柴棒数量，然后枚举A和B，计算C，检查火柴棒总数是否恰好为n。

💡 核心考点

枚举法、哈希表预处理。

🔎 执行追踪

火柴棒消耗表：0→6, 1→2, 2→5, 3→5, 4→4, 5→5, 6→6, 7→3, 8→7, 9→6。

对于n=14：去掉+和=各2根，剩10根给A、B、C。枚举A、B，计算match(A)+match(B)+match(A+B)=10的方案数。

⚠️ 易错提醒

注意A!=B时A+B=C和B+A=C算两个不同等式；A==B时只算一个。数字0也需要6根火柴棒。

```python
def match_count(x):
    """计算数字x需要的火柴棒数量"""
    matches = [6, 2, 5, 5, 4, 5, 6, 3, 7, 6]
    if x == 0:
        return 6
    count = 0
    while x > 0:
        count += matches[x % 10]
        x //= 10
    return count

n = int(input())
# 加号和等号共需要 2+2=4 根火柴棒
remain = n - 4
ans = 0

# 枚举 A 和 B，数字范围不会太大（最多 1111 左右）
for A in range(0, 1200):
    for B in range(0, 1200):
        C = A + B
        if match_count(A) + match_count(B) + match_count(C) == remain:
            if A != B:
                ans += 2  # A+B=C 和 B+A=C 算不同
            else:
                ans += 1

print(ans)
```

（13）Y3696 古籍名单解密

题目描述

小英是古籍数字化保护专业的学员，暑假期间获得了去国家图书馆数字化中心实习的机会。主任交给她一项任务——解密历史古籍中被加密过的一些传承人名单。经过研究，小英发现了如下加密规律（括号中是一个"原文 → 密文"的例子）：

1. 原文中所有的字符都在字母表中被循环左移了三个位置（`dec` → `abz`）。
2. 逆序存储（`abcd` → `dcba`）。
3. 大小写反转（`abXY` → `ABxy`）。

请你编写程序：输入加密后的字符串，输出解密后的传承人名单。

输入描述

一个加密的字符串。（长度小于 50 且只包含大小写字母）

输出描述

输出解密后的字符串。

输入样例 1

```
GSOOWFASOq
```

输出样例 1

```
Trvdizrrvj
```

【正确答案】见解析

【题目解析】

✅ 解题思路

本题是字符串处理题。解密是加密的逆过程：先逆序（还原逆序）→再循环右移3位（还原左移）→最后大小写反转（还原大小写反转）。

💡 核心考点

字符串处理、循环移位、逆序、大小写转换。

🔎 执行追踪

加密过程：原文→左移3位→逆序→大小写反转=密文。

解密过程：密文→大小写反转（还原）→逆序（还原）→右移3位（还原）。

以GSOOWFASOq为例：大小写反转→gsoowfasoQ→逆序→QOasfwoosg→右移3位→Trvdizrrvj。

⚠️ 易错提醒

注意解密顺序要与加密顺序严格相反。循环右移时注意边界（z→a→b→c的循环）。

```python
s = input()

# 步骤1：大小写反转
step1 = ''
for ch in s:
    if ch.isupper():
        step1 += ch.lower()
    else:
        step1 += ch.upper()

# 步骤2：逆序
step2 = step1[::-1]

# 步骤3：循环右移3位（还原左移3位）
result = ''
for ch in step2:
    if 'a' <= ch <= 'z':
        result += chr((ord(ch) - ord('a') + 3) % 26 + ord('a'))
    else:
        result += chr((ord(ch) - ord('A') + 3) % 26 + ord('A'))

print(result)
```

（14）Y3697 数字遗址寻路

题目描述

数字考古队员在古代遗址的数字地图中执行任务时不小心走入了一片复杂区域。该区域可以看成是由 $n \times n$ 的格点组成，每个格点只有 2 种状态，`.` 和 `#`，前者表示可通行，后者表示不可通行的障碍物。考古队员只能向上下左右四个方向之一移动。考古队员需要从 A 点赶到 B 点会合，问在不走出区域的情况下能不能办到。如果起点或者终点在障碍物中（为 `#`），则无法办到。

输入描述

第 1 行是测试数据的组数 k，后面跟着 k 组输入。每组测试数据的第 1 行是一个正整数 n（$1 \le n \le 100$），表示区域的规模是 $n \times n$ 的。接下来是一个 $n \times n$ 的矩阵，矩阵中的元素为 `.` 或者 `#`。再接下来一行是 4 个整数 ha、la、hb、lb，描述 A 处在第 ha 行、第 la 列，B 处在第 hb 行、第 lb 列。注意 ha、la、hb、lb 全部是从 0 开始计数的。

输出描述

k 行，每行输出对应一个输入。能办到则输出 `YES`，否则输出 `NO`。

输入样例 1

```
2
3
.##
..#
#..
0 0 2 2
5
.....
###.#
..#..
###..
...#.
0 0 4 0
```

输出样例 1

```
YES
NO
```

【正确答案】见解析

【题目解析】

✅ 解题思路

本题是标准的网格BFS/DFS连通性判断问题。使用BFS或DFS从起点出发搜索是否能到达终点。

💡 核心考点

BFS/DFS网格搜索、连通性判断。

🔎 执行追踪

从起点(ha,la)开始BFS/DFS，标记访问过的格子，如果遇到终点(hb,lb)则输出YES，搜索结束后未到达则输出NO。

⚠️ 易错提醒

注意先检查起点和终点是否为#，如果是直接输出NO。网格坐标从0开始计数。BFS用队列，DFS用递归或栈。

```python
from collections import deque

def bfs(grid, n, ha, la, hb, lb):
    if grid[ha][la] == '#' or grid[hb][lb] == '#':
        return False

    visited = [[False] * n for _ in range(n)]
    queue = deque()
    queue.append((ha, la))
    visited[ha][la] = True
    dx = [0, 0, 1, -1]
    dy = [1, -1, 0, 0]

    while queue:
        x, y = queue.popleft()
        if x == hb and y == lb:
            return True
        for i in range(4):
            nx, ny = x + dx[i], y + dy[i]
            if 0 <= nx < n and 0 <= ny < n and not visited[nx][ny] and gri
d[nx][ny] == '.':
                visited[nx][ny] = True
                queue.append((nx, ny))
    return False

k = int(input())
for _ in range(k):
    n = int(input())
    grid = []
    for _ in range(n):
        grid.append(input().strip())
    ha, la, hb, lb = map(int, input().split())
    if bfs(grid, n, ha, la, hb, lb):
        print("YES")
    else:
        print("NO")
```