INVITATION

物理解谜附件（纸面解谜）

你收到一份匿名“同行评审附件”。附件内容由四段物理记录与一段简单程序组成。
你的任务是从每段记录中计算出一个整数：A、B、C、D。
将 A、B、C、D 填入最后一页的解码器并运行，你将得到一个可访问的 URL。

规则
1）不需要联网；只需本地运行 Python 3。
2）允许使用计算器与草稿纸。
3）仅凭本附件即可解出答案（不需要外部资料）。

提示：四段记录分别对应：相空间与 Liouville、量子比特层析、Berry 相位、以及统计物理的“做账”问题。
每段的计算都只用到页面上提供的公式与数据。

记录 I：相空间审计（Liouville）

在“系综追踪”实验中，仪器记录了某个相空间密度峰值 rho_max(t)（已归一化，使 rho_max(0)=1）：

t (s)	0	1	2	3	4
rho_max(t)	1.00	1.35	1.82	2.46	3.32

旁注（来自一位很烦人的人）：
“若这是完整未筛选的哈密顿系综，Liouville 意味着局部体积不收缩；
若你看到系统性变尖，要么你看到的是筛选后的子系综，要么动力学根本不哈密顿。”

本页任务
1）用模型 rho_max(t) ≈ exp(gamma * t) 估计 gamma（单位 s^-1）。
2）定义：A = round(1000 * gamma)（round 为四舍五入到整数）。

请填写： A = _____

记录 II：单量子比特“简化层析”

同一未知量子比特态 rho 被重复制备并分别测量 sigma_x、sigma_y、sigma_z。得到测量结果为 +1 的频率：

测 sigma_x：p_x(+)=0.80
测 sigma_y：p_y(+)=0.35
测 sigma_z：p_z(+)=0.60

给定（可直接使用）：
p_i(+)= (1 + r_i)/2 ——> r_i = 2*p_i(+) − 1
其中 r = (r_x, r_y, r_z) 是 Bloch 向量。

本页任务
定义：B = round(100 * (r_x − r_y + r_z))

请填写： B = _____

记录 III：Berry 相位“指纹”

自旋 1/2 粒子在缓慢变化的磁场中做绝热闭合循环。磁场方向在单位球面上围成固角：

Omega = pi/2

给定（可直接使用）：
gamma_B = − Omega / 2 （相位按 2*pi 取模）

本页任务
先把 gamma_B 取到区间 [0, 2*pi) 的等价值，再令：
C = round(1000 * frac(gamma_B / pi))
其中 frac 表示小数部分（例如 1.75 的 frac 是 0.75）。

请填写： C = _____

记录 IV：熵账本审计（“妖”的漏洞）

账本声称：

“系统完全隔离、严格哈密顿演化。”

初态：能量壳可达微观态数 W0 = 10^12。

10 秒后：可达微观态数 W1 = 10^9。

并写道：“因此熵减 DeltaS = k*ln(W1/W0)，实现无代价制冷。”

但夹页里还有一条小字（非常关键）：

“注意：最终统计只包含通过门控条件的轨道 – 在 t=10s 时刻落入探测窗 R 的样本才被记录，其余样本直接丢弃。”

本页任务

判断：账本的“熵减叙述”究竟靠什么手段“做账”才成立？请选择唯一最贴切的一项，并将选项编号记为 D：

1）真正的耗散/摩擦导致相空间体积收缩（非哈密顿）。

2）后验筛选/条件化（post-selection）：只保留满足门控的子系综。

3）只是粗粒化熵的变化，与 W 的含义无关；账本没错。

4）初末不在同一能量壳：能量漂移使 W 变化，但仍隔离哈密顿。

请填写： D = _____

解码器：简单编程

你现在应已得到整数 A、B、C、D。把它们填入下列程序并运行。程序输出即为最终答案（URL）。

密文（十六进制字符串）：

```
4a10f1e3306dfa7bc0a089d092df6552329eaf6a56866beee98d09bbf35c7bd38d8e09
```

Python 3 程序（可直接复制运行）：

```
# === Fill in your computed A, B, C, D ===
A = None
B = None
C = None
D = None



cipher_hex = "4a10f1e3306dfa7bc0a089d092df6552329eaf6a56866beee98d09bbf35c7bd38d8e09"
cipher = bytes.fromhex(cipher_hex)

def xs32(x: int) -> int:
    x &= 0xFFFFFFFF
    x ^= (x << 13) & 0xFFFFFFFF
    x ^= (x >> 17) & 0xFFFFFFFF
    x ^= (x << 5)  & 0xFFFFFFFF
    return x & 0xFFFFFFFF

seed = (A << 20) | (B << 12) | (C << 2) | D

x = seed
out = bytearray()
for b in cipher:
    x = xs32(x)
    out.append(b ^ (x & 0xFF))

print(out.decode("ascii"))
```

如果你想确认自己没有算错：先检查校验式是否成立，再运行解码器。
祝你解出邀请函。