T1 季风

这道题本质是一个**简单的分类判断题**，核心就是根据两个数的正负来确定方向。

一、解题思路

题目给了两个量：

- (a)：南北方向
- (b)：东西方向

我们只需要关注**正负号**：

- (a > 0) → 北（North）
- (a < 0) → 南（South）
- (b > 0) → 东（East）
- (b < 0) → 西（West）

然后把两个方向**拼接起来**即可。

判断逻辑（核心）

可以分成两步：

1️⃣ 先判断南北方向

```
if(a > 0) → "North"
else      → "South"
```

2️⃣ 再判断东西方向

```
if(b > 0) → "East"
else      → "West"
```

3️⃣ 拼接输出

```
cout << 南北 + 东西;
```

二、示例分析

输入：

```
-20 20
```

- (a < 0) → South
- (b > 0) → East

👉 结果： `SouthEast`

三、考察知识点

这题主要考察非常基础的内容：

1. **条件判断（if-else）**
2. **正负数判断**
3. **字符串拼接**
4. **输入输出**

四、一句话总结

👉 **根据 a、b 的正负分别确定南北和东西方向，然后拼接成最终答案。**

T2 购票

这道题本质是一个**取最小值的简单决策问题**。

一、解题思路

你只有两种买票方式：

1️⃣ 全部买单次票

总花费：$n \times a$

2️⃣ 买一张年票

总花费：b

✅ 最终答案

直接取两者的最小值即可：$\min(n \times a,; b)$

二、关键细节

⚠️ 注意 1：n 可能为 0

- 如果 (n = 0)，不去公园
- 花费 = **0**

👉 所以可以特判，也可以让公式自然成立（0 × a = 0）

⚠️ 注意 2：会爆 `int`

数据范围：

- ($n, a, b \le 10^{18}$)

👉 ($n \times a$) 可能达到：$10^{18} \times 10^{18} = 10^{36}$

所以必须使用：

- C++：`long long`（但本题其实可能要用 `__int128` 更安全）

三、示例理解

输入：

```
2 10 50
```

- 单次票：$(2 \times 10 = 20)$
- 年票：50

👉 选更小 → `20`

四、考察知识点

这题主要考察：

1. **简单数学建模**
   - 把现实问题转化为公式： `min(n*a, b)`
2. **最小值比较**
   - 使用 `min()`
3. **大整数处理**
   - 防止溢出（ `long long / __int128` ）
4. **边界情况处理**
   - (n = 0)

五、一句话总结

👉 **要么按次买，要么直接买年票，取更便宜的那一个即可。**

T3 百万富翁

这道题本质是一个**顺序模拟 + 条件停止**的问题。

一、解题思路

从第 1 台机器开始，**按顺序模拟整个过程**即可。

设当前积分为 `cur` （初始为 (x)）。

🔁 对每一台抽奖机做如下操作：

1️⃣ 能不能玩？

在第 i 台之前判断：

- 如果 $cur < a_i$
  👉 钱不够，**直接停止**

2️⃣ 可以玩 → 更新积分

- 花费：$-a_i$

- 获得：$+b_i$

👉 更新：$cur = cur - a_i + b_i$

3️⃣ 是否达到目标?

- 如果 $cur \geq y$
  👉 **立即停止**

二、流程总结（核心逻辑）

```
for 每台机器 i
    if cur < a[i]
        break
    cur = cur - a[i] + b[i]
    if cur >= y
        break
```

三、关键点（很重要）

✅ 1. 顺序不能乱

必须按 **1 → n** 顺序，不能贪心选机器

✅ 2. 两种停止情况

- ❌ 钱不够 → `cur < a[i]`
- ✅ 达到目标 → `cur >= y`

✅ 3. 不要提前判断 y

👉 是“玩完之后”才判断是否 ≥ y

四、考察知识点

这题重点很基础但很典型：

1. **模拟（Simulation）**
   - 按题意一步步执行

2. **循环控制**
    - `for + break`
3. **条件判断**
    - 两种停止条件
4. **顺序处理思想**
    - 不能打乱顺序（不是贪心 / 排序题）

五、一句话总结

👉 **从前往后模拟，每次更新积分，遇到“钱不够”或“达到目标”就停止。**

T4 卡牌游戏

这道题本质是一个**模拟 + 双指针（或队列）**的问题，思路很清晰。

一、解题思路

1️⃣ 先分成两堆

- A 堆：前 (k) 张 → $a[1] \sim a[k]$
- B 堆：后面的 → $a[k+1] \sim a[n]$

2️⃣ 轮流取牌（核心）

规则是：

- **先从 A 取一张**
- 再从 B 取一张
- 一直重复

👉 相当于“交替合并两个序列”

3️⃣ 什么时候停止交替？

当 **A 或 B 有一个取完** 时停止交替

4️⃣ 把剩下的直接接上

- 如果 A 还有剩 → 全部接到后面
- 如果 B 还有剩 → 全部接到后面

二、实现方法（推荐）

用两个指针模拟最简单：

- `i` 指向 A（从 1 开始）
- `j` 指向 B（从 k+1 开始）

核心流程

```
i = 1, j = k+1

while (i <= k && j <= n):
    放 a[i]
    i++
    放 a[j]
    j++

把剩余的 A 或 B 全部加入
```

三、示例理解

输入：

```
n = 6, k = 2
1 2 | 3 4 5 6
```

过程：

- A：1 2
- B：3 4 5 6

交替：

- 1, 3

- 2, 4

A 用完 → 剩下 B：

- 5, 6

👉 最终：

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

四、特殊情况

✅ k = 1

- A 只有一个
- 取一次后直接接 B

✅ k = n

- B 为空
- 结果就是原数组（不变）

五、考察知识点

这题主要考察：

1. **模拟（Simulation）**
2. **双指针**
3. **数组操作**
4. **顺序处理逻辑**

六、一句话总结

👉 **把数组分成两段，然后像“拉链一样交替合并”，最后接上剩余部分。**

T5 翻转和反转

这道题表面是**字符串操作模拟**，但如果每次都真的去翻转/取反，会超时。
👉 正确思路是：**用“标记（懒操作）”优化**。

一、核心思路（重点！）

我们发现两种操作：

1️⃣ 翻转（reverse）

把字符串倒过来
👉 本质：**顺序改变**

2️⃣ 反转（0 ↔ 1）

👉 本质：**值改变**

❗关键优化思想

不用每次真的操作字符串，而是用两个变量记录状态：

- `rev`：是否被翻转过（0 / 1）
- `flip`：是否被取反过（0 / 1）

二、如何处理操作?

遍历操作串 `w`：

- 如果是 `'1'`（翻转）
  👉 `rev ^= 1`
- 如果是 `'2'`（取反）
  👉 `flip ^= 1`

三、最后如何得到答案?

1️⃣ 先确定遍历方向

- 如果 `rev == 0` → 正序
- 如果 `rev == 1` → 倒序

2️⃣ 再决定是否取反

- 如果 `flip == 1` → 每一位取反

四、整体流程

```
读入 n, q
读入字符串 s
读入操作串 w

rev = 0, flip = 0

for 每个操作:
    if '1': rev ^= 1
    if '2': flip ^= 1

如果 rev == 0:
    从左到右输出
否则:
    从右到左输出

输出时:
    如果 flip == 1 → 字符取反
```

五、举个简单例子

```
s = 1001
操作：12
```

- 操作1（翻转）→ rev=1
- 操作2（取反）→ flip=1

最终：

- 倒序：1001 → 1001
- 取反：0110

六、为什么这样做？

因为：

- 翻转做两次 = 没做（异或）
- 取反做两次 = 没做（异或）

👉 所以可以用 **状态压缩**

七、考察知识点

1. **字符串处理**
2. **模拟优化**
3. **状态压缩（懒标记思想）**
4. **异或运算（^）**
5. **双端遍历（正序 / 逆序）**

八、一句话总结

👉 **不用真的改字符串，只记录“是否翻转、是否取反”，最后一次性输出即可。**

T6 影子字符串

这道题本质是一个**去重 + 保留首次出现顺序**的问题。

一、解题思路

✅ 什么是“影子字符串”？

- 如果一个字符串**之前已经出现过**
- 那么它后面再出现的都是“影子”

👉 只保留**第一次出现的那一次**

二、核心做法

1️⃣ 逐行读入字符串

- 遇到 `"0"` 结束

2️⃣ 用一个集合记录是否出现过

比如用：

- 使用字符串数组进行存储字符串，整数数组计数。

3️⃣ 判断逻辑

```
如果这个字符串 没出现过：
    加入答案
    标记为已出现
否则：
    跳过（影子）
```

4️⃣ 拼接结果

- 按输入顺序直接拼接即可
-

四、注意点

⚠️ 1. 输入结束条件

- 遇到 `"0"` 就停止（不是数字 0，而是字符串）

⚠️ 2. 不能排序

- 必须保持**原输入顺序**

五、考察知识点

1. **字符串处理**

2. **去重 + 保序**
3. **输入控制（读到特定标志结束）**

六、一句话总结

👉 **用数组标记去重，只保留第一次出现的字符串，按顺序拼接输出。**