高级程序设计与实验随堂测试十一

1. (16 分)　**题目描述：**

作为授课教师，你需要从题库中挑选 K 道题目组成一份难度精准的试卷。题库中有 N 道备选题目，每道题有两个属性：难度值 diff 和考察知识点编号 category。题目信息以 vector<vector<int>> questions 给出，其中 questions[i] = {diff, category}。

为了保证试卷质量，需要同时满足：

1. 选出的 K 道题难度值总和**恰好**等于给定的 target_diff。
2. 任意两道题的知识点类别 category **不能相同**（即考察面要广）。

请写一个 C++函数，打印所有符合条件的抽题组合列表（返回题目在原题库中的索引即可）。

函数格式：

void choose(const vector<vector<int>> & questions, int K, int target_diff);

你的函数在做搜索时应当尽量早地开始剪枝；如果枚举出所有组合再确认其满足所求条件，会浪费大量计算时间，最多得 12 分。

输入示例 (Input)

- **K (需抽取的题目数量):** 3
- **target_diff (目标总难度值):** 14
- **questions (题库列表):**

[

[3, 1], // 索引 0：难度 3，知识点 1

[4, 2], // 索引 1：难度 4，知识点 2

[4, 3], // 索引 2：难度 4，知识点 3

[5, 1], // 索引 3：难度 5，知识点 1

[6, 4], // 索引 4：难度 6，知识点 4

[7, 5] // 索引 5：难度 7，知识点 5

]

输出示例：

0, 1, 5

0, 2, 5

1, 2, 4

说明：

[0, 3, 4]： 对应难度 3 + 5 + 6 = 14。但是，索引 0 和索引 3 的知识点**都是 1**

2.（14 分）**题目描述：**

一个大型分布式系统中有 N 个微服务需要依次冷启动（必须全部启动，且不能并发，所以一个启动顺序是这 N 个服务的一个全排列）。 每次启动一个服务，系统会加载相关的底层库。如果紧接着启动的下一个服务刚好依赖相似的底层库，就能产生“缓存命中增益”，从而缩短总启动时间。

给定一个 N × N 的二维数组 vector<vector<int>> bonus，其中 bonus[i][j] 表示在启动微服务 i 后**紧接着**启动微服务 j 所能节省的时间（注意：bonus[i][j] 与 bonus[j][i] 不一定相等）。写一个 C++函数，找出能使**总增益最大化**的启动序列，打印这个序列，以及它的最大增益值。你的函数既可以通过递归枚举所有的启动序列，也可以使用更高效的动态规划。

函数格式：`void sequence(int N, const vector<vector<int>> & bonus);`

输入示例 (Input)

- **N (微服务总数):** 4 (编号为 0, 1, 2, 3)
- **bonus (缓存增益矩阵):**

```
vector<vector<int>> bonus = {
 // 0   1   2   3   (紧接着启动的服务 j)
 { 0,  2, 10,  1 }, // 启动 0 后
 { 5,  0,  4, 20 }, // 启动 1 后
 { 3, 15,  0,  6 }, // 启动 2 后
 { 8,  7,  9,  0 } // 启动 3 后
};
```

输出示例 (Output)

```
最大缓存增益总和 (Max Bonus): 45
最优启动序列 (Optimal Sequence): [0, 2, 1, 3]
```