4-3：

> *若在0-1 背包问题中，各物品依重量递增排列时，其价值恰好依据减序排列。对这个特殊的0-1背包问题，设计一个有效算法找出最优解，并说明算法的正确性。*

题目分析

0-1 背包问题在一般情况下是一个无法使用“贪心”算法求解的问题，但因其具有最优子结构性质，因此可以使用动态规划求解。但对于本题，由于多了两个限制，“各物品依重量递增排列时，其价值恰好依据减序排列”，因此本题是一个特殊的 0-1 背包问题，可以使用贪心算法求解。贪心思想为：每次都往背包里放重量小的物品，直到背包装不下为止。

现已知物品数量为 n，背包最大承重为 C，w_i 表示编号为 i 的物品的重量，v_i 表示编号为 i 的物品的价值。下面证明贪心算法的正确性：

设通过贪心算法求出来的一组解向量为 $S(k)=\{x_1,x_2,\cdots,x_k\}$，最优解为 $T(m)=\{b_1,b_2,\cdots,b_m\}$，可以证明，若最优解唯一，则 $S(k)=T(m)$。

反证：假设 $S(k)$ 不是最优解，则 $S(k)$ 与 $T(m)$ 中必存在不相同的元素，设这一个元素为 x_i，有如下几种情况：

- 最优解选取 x_i，贪心算法构造的解不选取 x_i，由于贪心算法的构建是从重量小往重量大的依次选取，所以若不选取 x_i，说明背包已经无法承重；而对于 $x_p(p<i)$，最优解与贪心解一致，因此最优解也会因无法承重而无法选取 x_i，矛盾。
- 最优解不选取 x_i，贪心解选取 x_i，由于最优解要价值最优，因此必然会再选取后续的 $x_k(k>i)$，而由于重量递增，价值递减，若将最优解中的 x_k 以 x_i 替代，则构造出的新解比原最优解更优，矛盾。

综上，贪心算法构造出的贪心解即最优解。

代码实现

在代码实现中，为方便起见，省去排序过程（否则需要使用结构体并进行排序），可以认为排序使用复杂度恒为 $O(nlogn)$ 的归并排序。

代码实现详见4-3 Code.cpp

输出示例

规定输入格式如下，共输入三行：

- 第一行两个正整数 n,C，n 表示物品的数量，C 表示背包的最大承重。
- 第二行 n 个正整数 w_i，表示编号为 i 的物品的重量
- 第三行 n 个正整数 v_i，表示编号为 i 的物品的价值

```
输入1：
10 15
```

```
1 3 5 7 9 11 13 15 17 19
20 18 16 14 12 10 8 6 4 2
输出1：
选取的物品编号为：1 2 3
背包最大价值为：54

输入2：
2 3
1 3
5 2
输出2：
选取的物品编号为：1
背包最大价值为：5

输入3：
3 5
6 7 8
999 8 7
输出3：
选取的物品编号为：不选取物品。
背包最大价值为：0
```

算法分析

本算法所涉及的部分如下：

- 输入输出：需要 $O(N)$ 复杂度。
- 选取物品：需要 $O(N)$ 复杂度。
- 将物品按重量递增或价值递减顺序排序：需要 $O(NlogN)$ 复杂度。

综上，本算法的复杂度为 $O(NlogN)$。