实验三

编程题 1 （总分：100.00）

#	题目	分值
1	**实验3-1: 0-1 Knapsack**	100.00

Description

0-1 背包问题描述如下：给定n 种物品和一个背包。物品i的重量是w i，其价值为vi ，背包的容量为C。应如何选择装入背包的物品，使得装入背包中物品的总价值最大? 在选择装入背包的物品时，对每种物品i只有2 种选择，即装入背包或不装入背包。不能将物品i 装入背包多次，也不能只装入部分的物品i。

Input

输入由多组测试数据组成。
每组测试数据输入的第一行有2个正整数n和c。n是物品数，c是背包的容量。接下来的1 行中有n个正整数，表示物品的价值。第3 行中有n个正整数，表示物品的重量。

Output

对应每组输入，输出的2行是装入背包物品的最大价值和最优装入方案。

Sample Input

5 10

6 3 5 4 6

2 2 6 5 4

Sample Output

15

1 1 0 0 1

参考答案:

```
#include <iostream>
#include <iomanip>
#include <string>

int min(int w, int c) {
    int temp;
    if (w < c) temp = w;
    else temp = c;
    return temp;
}

int max(int w, int c) {
    int temp;
    if (w > c) temp = w;
    else temp = c;
    return temp;
}

void knapsack(int v[], int w[], int c, int n, int** m) {
    int jmax = min(w[n] - 1, c);
    for (int j = 0; j <= jmax; j++)
        m[n][j] = 0;
    for (int jj = w[n]; jj <= c; jj++)
        m[n][jj] = v[n];
    for (int i = n - 1; i > 1; i--) {
        jmax = min(w[i] - 1, c);
        for (int j = 0; j <= jmax; j++)
            m[i][j] = m[i + 1][j];
        for (int jj = w[i]; jj <= c; jj++)
            m[i][jj] = max(m[i + 1][jj], m[i + 1][jj - w[
i]] + v[i]);
    }
    m[1][c] = m[2][c];
    if (c >= w[1])
        m[1][c] = max(m[1][c], m[2][c - w[1]] + v[1]);
    std::cout << m[1][c] << std::endl;
}

int traceback(int **m, int w[], int c, int n, int x[]) {
    for (int i = 1; i < n; i++) {
        if (m[i][c] == m[i + 1][c])
            x[i] = 0;
        else {
            x[i] = 1;
            c -= w[i];
        }
    }
    x[n] = (m[n][c]) ? 1 : 0;
    for (int y = 1; y <= n; y++) {
        std::cout << x[y] << " ";
```

```
    }
    std::cout << std::endl;
    return x[n];
}

int main() {
    int n, c;
    int **m;
    std::cin >> n >> c;
    int *v = new int[n + 1];
    for (int i = 1; i <= n; i++)
        std::cin >> v[i];
    int *w = new int[n + 1];
    for (int j = 1; j <= n; j++)
        std::cin >> w[j];
    int *x = new int[n + 1];
    m = new int*[n + 1];
    for (int p = 0; p < n + 1; p++) {
        m[p] = new int[c + 1];
    }
    knapsack(v, w, c, n, m);
    traceback(m, w, c, n, x);
    delete[] x;
    delete[] w;
    delete[] v;
    return 0;
}
```