[课后作业 1](#_Toc96527831)

[**1.** 两数之和问题的另一种解法 1](#_Toc96527832)

[**2.** 回文判断 1](#_Toc96527833)

[**3.** 删除重复元素 2](#_Toc96527834)

[**4.** 定差数对 2](#_Toc96527835)

[**5.** 多重集交 2](#_Toc96527836)

[**6.** (USACO Training - Section 1.3 Problem 1) Milking Cows 3](#_Toc96527837)

[**7.** (USACO 2018 January – Bronze 3) Out of Place 4](#_Toc96527838)

课后作业

1. 两数之和问题的另一种解法

给定（）个整数（）。给定（），求所有数对中和为的对数。

输入样例：

|  |
| --- |
| 6 10  1 2 3 4 6 9 |

输出样例：

|  |
| --- |
| 3 |

证明以下程序的正确性：

|  |
| --- |
| 1. int solve(const vector<int> &a, int m) { 2. int ans = 0; 3. for (int i = 0, j = (int)a.size() - 1; i < j; i++) { 4. while (j > i && a[i] + a[j] > m) { 5. j--; 6. } 7. if (j > i && a[i] + a[j] == m) { 8. ans++; 9. } 10. } 11. return ans; 12. } |

1. 回文判断

编写一个函数，接收一个字符串作为参数。判断是否是回文字符串。中可能包含任意非控制字符。判断回文时仅考虑字母和数字，并且不考虑大小写。

|  |
| --- |
| bool check\_palindrome(const string &s) |

输入样例：

|  |
| --- |
| Do geese see God? |

输出样例：

|  |
| --- |
| Yes |

本题的限制为：你的函数需要有的运行时间和的额外运行空间。

1. 删除重复元素

编写一个函数，接收一个排序数组作为参数，删除其中的重复元素，并返回剩余元素的数量。

|  |
| --- |
| int remove\_duplicates(vector<int> &a) |

输入样例：

|  |
| --- |
| 8  1 2 2 4 4 4 4 8 |

输出样例：

|  |
| --- |
| 4  1 2 4 8 |

本题的限制为：函数需要有的时间复杂度及的额外空间复杂度。

1. 定差数对

给定（）个各不相同的整数（）。给定（），求所有数对中差为的对数。

输入样例：

|  |
| --- |
| 5 3  1 2 3 5 8 |

输出样例：

|  |
| --- |
| 2 |

1. 多重集交

给定两个多重集（同一元素可以出现多次的集合），求这两个多重集的交的元素数量。某一个数在两个多重集的交之中的出现次数等于其在两个多重集之中出现次数的较小值。第一行输入（），第二行输入第一个多重集中的所有元素（），第三行输入第二个多重集中的所有元素（）。输出两多重集之交的元素数量。

输入样例：

|  |
| --- |
| 6 6  1 2 1 3 1 4  1 1 2 3 5 8 |

输出样例：

|  |
| --- |
| 4 |

1. (USACO Training - Section 1.3 Problem 1) Milking Cows

Three farmers rise at 5 am each morning and head for the barn to milk three cows. The first farmer begins milking his cow at time 300 (measured in seconds after 5 am) and ends at time 1000. The second farmer begins at time 700 and ends at time 1200. The third farmer begins at time 1500 and ends at time 2100. The longest continuous time during which at least one farmer was milking a cow was 900 seconds (from 300 to 1200). The longest time no milking was done, between the beginning and the ending of all milking, was 300 seconds (1500 minus 1200).

Your job is to write a program that will examine a list of beginning and ending times for N (1 <= N <= 5000) farmers milking N cows and compute (in seconds):

* The longest time interval at least one cow was milked.
* The longest time interval (after milking starts) during which no cows were being milked.

**NOTE:** Milking from time 1 through 10, then from time 11 through 20 counts as two different time intervals.

**PROGRAM NAME: milk2**

**INPUT FORMAT**

Line 1: The single integer, N

Lines 2..N+1: Two non-negative integers less than 1,000,000, respectively the starting and ending time in seconds after 0500

**SAMPLE INPUT (file milk2.in)**

|  |
| --- |
| 3  300 1000  700 1200  1500 2100 |

**OUTPUT FORMAT**

A single line with two integers that represent the longest continuous time of milking and the longest idle time.

**SAMPLE OUTPUT (file milk2.out)**

|  |
| --- |
| 900 300 |

1. (USACO 2018 January – Bronze 3) Out of Place

Feeling ambitious, Farmer John plans to attempt something that never seems to go quite right: he wants to take a photograph of his entire herd of cows.

To make the photograph look nice, he wants the cows to line up in a single row from shortest to tallest. Unfortunately, right after he has the cows line up this way, Bessie the cow, always the troublemaker, steps out of line and re-inserts herself at some other location in the lineup!

Farmer John would like to swap pairs of cows so the entire herd is again lined up properly. Please help him determine the minimum number of swaps he needs to make between pairs of cows in order to achieve this goal.

**INPUT FORMAT (file outofplace.in):**

The first line of input contains (). The next lines describe the heights of the cows as they are lined up after Bessie makes her move. Each cow height is an integer in the range . Cows may have the same height.

**OUTPUT FORMAT (file outofplace.out):**

Please output the minimum number of times Farmer John needs to swap pairs of cows in order to achieve a proper ordering. Swaps do not necessarily need to involve adjacent cows in the ordering.

**SAMPLE INPUT:**

|  |
| --- |
| 6  2  4  7  7  9  3 |

**SAMPLE OUTPUT:**

|  |
| --- |
| 3 |

In this example, Bessie is clearly the cow of height 3. FJ return the cows to sorted order using three swaps as described below:

|  |
| --- |
| 2 4 7 7 9 3 - Original Lineup  2 4 7 7 3 9 - Swap the last two cows  2 4 3 7 7 9 - Swap the first 7 and 3  2 3 4 7 7 9 - Swap 4 and 3 |

Problem credits: Brian Dean