Queen Mary
University of London
School of Electronic Engineering and Computer Science

7th EECS Programming Competition

Sponsored by FDM

Wednesday, 12 February 2014

Note. Your programs should read from **standard input**. The programs should process one input and print the result at the **standard output**.

When testing your programs we will run them several times on different inputs. In order to get a point for that problem your program must terminate within 5 seconds outputing the correct solution on each of the input tests. Programs should not use more than 2MB of memory to solve a problem.

A. Bubble Sort

Problem. Bubble sort is one of the simplest sorting algorithms. Given a list of numbers, e.g.

5 2 3 1 7 6

the algorithm works by repeatedly scanning the list from left to right and swapping any two adjacent numbers which are not in the right (increasing) order. For instance, in the first scan on the list above one would swap 5 and 2, 5 and 3, 5 and 1, and 7 and 6, to obtain the list

2 3 1 5 6 7

In the second scan one would then swap 3 and 1, to obtain the list

2 1 3 5 6 7

and in the final scan 2 and 1 are swapped to obtain the ordered list. After a scan where no swaps are necessary the algorithm terminates. You are asked to write a program that counts the number of scans, and the total number of swaps required to bubble sort a given list of numbers.

Input. The input consists of a number $1 < N < 216$, indicating the length of the given list, followed by N numbers. The N numbers are in the range 0 to $2^{32} - 1$, and might contain repeated numbers.

Output. Your program should output (in the format shown below) the number of scans and the total number of swaps needed to bubble sort the given list. The final scan where no swaps are performed should not be counted.

Sample Inputs.

Sample Input 1	**Output to Sample Input 1**

`4 4 4 1 5`	`Scans: 2`
	`Total swaps: 2`

Sample Input 2	**Output to Sample Input 2**

`6 5 2 3 1 7 6`	`Scans: 3`
	`Total swaps: 6`

B. Dominos

Problem. Domino tiles contain two numbers from the set $\{0, 1, 2, 3, 4, 5, 6\}$. Two tiles can be put together if they have a common number, as in the picture below.

Given a set of domino tiles, you are asked to write a program that finds the length of the longest sequence one can form with such tiles. For this problem each *title type* will be described as $[x{:}y]$, where x and y are numbers in the set $\{0, 1, 2, 3, 4, 5, 6\}$. For instance, the title containing 1 and 6 can be represented as either [1:6] or [6:1]. For each title type *you will also be told how many* of these are available.

Input. The input starts with a number $N \geq 1$ describing how may tile types are available. After that it follows N lines describing the title types and how many of each are available. The number of available tiles of each type will be at most 10, i.e. a number in the range $1 \leq \ldots \leq 10$. Each tile type will be listed at most once. For instance, in the Sample Input 1 below one has two [1:5] tiles and one [1:1] title, meaning one can form a sequence of length 3.

Output. Length of longest possible sequence one can form with the given domino tiles.

Sample Inputs.

Sample Input 1	**Output to Sample Input 1**

`2`
`[1:5] 2`
`[1:1] 1`
	`3`

Sample Input 2	**Output to Sample Input 2**

`2`
`[3:2] 10`
`[6:6] 5`
	`10`

C. Phonebook

Problem. Given a list of people in a phonebook, you are asked to sort this list by their date of birth, from oldest to youngest.

Input. The input will consist of a number N (in the range 1 to 1000) followed by a list of N people, together with their date of birth. The name of each person might consist of several names separated by a space, but each name will contain only alpha characters, i.e. $\{$ A, B, ..., Z $\}$ or $\{$ a, b, ..., z $\}$. The name is followed immediately by a comma and a space, followed then by the date in the format DD/MM/YYYY. All months and days of the month will consist of precisely two digits, hence, for days or months in the range 1 to 9, these will be written as 01, 02, ...

Output. The given list sorted by date of birth, from oldest to youngest. When two persons where born on the same date, they should then be ordered lexicographically by last name. If a person is only listed with one name (for instance, "John") this is to be considered as the last name. When two persons where born on the same date and have the same last name, these can be listed in any order.

Sample Inputs.

Sample Input 1	**Output to Sample Input 1**

`3`
`John, 02/03/2001`
`Barbara, 01/01/1978`
`Peter, 23/07/1990`
	`Barbara, 01/01/1978`
	`Peter, 23/07/1990`
	`John, 02/03/2001`

Sample Input 2	**Output to Sample Input 2**

`5`
`Elizabeth, 15/07/2013`
`Ana Smith, 10/01/2013`
`John Scott, 02/03/2001`
`Barbara Kennedy, 10/01/1978`
`Peter Johnson, 02/03/2001`
	`Barbara Kennedy, 10/01/1978`
	`Peter Johnson, 02/03/2001`
	`John Scott, 02/03/2001`
	`Ana Smith, 10/01/2013`
	`Elizabeth, 15/07/2013`

D. Containers

Problem. Given a list of rectangular containers (dimensions given as $X \times Y$) find the maximum number of containers that can be stacked up inside each other. Note that a container with dimensions $X_1 \times Y_1$ can only be stacked up inside another container with dimensions $X_2 \times Y_2$ if $X_1 < X_2$ and $Y_1 < Y_2$, or $X_1 < Y_2$ and $X_2 < Y_1$. For instance, a container with dimensions 2.10×3.50 can be placed inside a container with dimensions 4.00×2.20. The height of the containers is not important.

Input. A number $1 < N \leq 100$ followed by a list of N containers. Each container is given by its dimension X and Y (in the format shown below) where X and Y are always given with two decimal places of precision.

Output. Length of longest sequence of containers that can be stacked up inside each other.

Sample Inputs.

Sample Input 1	Output to Sample Input 1

`2`
`2.10 3.50`
`4.00 2.20`
	`2`

Sample Input 2	Output to Sample Input 2

`5`
`1.00 1.00`
`2.00 2.00`
`1.10 1.90`
`1.15 1.95`
`1.20 1.80`
	`4`

E. DNA

Problem. The DNA molecule is formed from four so-called nucleobases, normally identified as G, A, T, C. It is known that "A" should bind with "T"; and that "G" should bind with "C". When a different binding takes place something horribly wrong might happen.

Input. The input will consist of two sequences of the characters from the set of four letters { G, A, T, C }, with corresponding nuclebases vertically aligned. This sequence of base pairs will have length at most 5000.

Output. Your program should identify whether any anomalies occurs in this section of a DNA. Hence, your program should Output either "No error detected" or "First error at position X", where X is the index of the first position where a mistake occurs. The indexing of positions start at 1.

Sample Inputs.

Sample Input 1	**Output to Sample Input 1**
`AGATGACCAG` `TCTATTGGTC`	`First error at position 5`

Sample Input 1	**Output to Sample Input 1**
`AGATGACCAGTTAG` `TCTACTGGTCAATC`	`No error detected`

F. Digits

Problem. The digit sum of a number is the sum of its digits when represented in ordinary base 10. Digit sums often appear in simple number puzzles but also have serious uses, such as divisibility testing and checksums.

Given a nonnegative integer, your task is to compute its digit sum.

Input. An integer N in the range $0 \leq N < 2^{32}$.

Output. The digit sum of N.

Sample Inputs.

Sample Input 1	**Output to Sample Input 1**
`42`	`6`

Sample Input 2	**Output to Sample Input 2**
`123456789`	`45`