Airplanes

Submission deadline:	**2022-11-28 11:59:59**	1377642.155 sec
Late submission with malus:	**2022-12-31 23:59:59** (Late submission malus: 100.0000 %)	
Evaluation:	**0.0000**	
Max. assessment:	**5.0000** (Without bonus points)	
Submissions:	0 / 20 Free retries + 10 Penalized retries (-10 % penalty each retry)	
Advices:	0 / 2 Advices for free + 2 Advices with a penalty (-10 % penalty each advice)	

The task is to develop a program to help airplane navigation and avoid collisions.

When operating an airport, it is important to avoid airplane collisions. This is guaranteed by the control tower, which uses a radar and a control software. Your task is to develop the software which receives radar data and detects possible airplane collisions, i.e., it finds the minimum distance among the airplanes reported by the radar.

The input of your program is the list of airplanes. We assume a simplification where each airplane is represented by a 2-D coordinate - `x` and `y` - and its name. Both `x` and `y` are decimal numbers. The name of the airplane is a string, the string does not contain whitespace characters and is at most 199 characters long. Airplane names are not unique, i.e., the same name may appear many times in the input. The number of airplanes may be very high, moreover, the total number of airplanes is not known beforehand. The input ends when EOF is reached. The input format is shown in the sample runs below.

The output of the program is the minimum distance found. Moreover, the program counts the number of airplane pairs where the distance is equal to the minimum distance and displays the planes. The (possibly) colliding airplanes are printed in pairs on separate lines, plane names are separated by a dash. The order of output lines in the listing as well as the order of names on a single output line is not specified. The testing environment re-arranges the order before it starts the comparison.

The program must scrutinize input data. If an invalid input is detected, the program must display an error message and terminate. The following is considered invalid input:

- non-numeric coordinate (not a valid floating point number),
- an extra or missing separator (comma, colon),
- the total number of airplanes less than 2 (at least two airplanes are required to compute some distance).

Your program will be tested in a restricted environment. The testing environment limits running time and available memory. The exact time and memory limits are shown in the reference solution testing log. The program is not memory greedy. However, the program may require a lot of time to compute the result if the number of airplanes is high. A reasonable implementation of the naive algorithm passes all mandatory test. However, it will not pass the bonus test, thus it will awarded be nominal 100%. An improved algorithm is required to pass the bonus test.

Sample program runs:

```
Plane coordinates:
0,0: KLM
5,0: Lufthansa
10,0: SmartWings
7,0: AirFrance
2,0: Qantas
Minimum airplane distance: 2.000000
Pairs found: 2
KLM - Qantas
Lufthansa - AirFrance
```

```
Plane coordinates:
0,5: Qantas
```

```
5,0: KLM
0,0: AirFrance
5,5: Lufthansa
2.5,2.5: KLM
Minimum airplane distance: 3.535534
Pairs found: 4
Qantas - KLM
AirFrance - KLM
KLM - KLM
KLM - Lufthansa
```

```
Plane coordinates:
-10,-5: Ryanair
10,0: LOT
12,12: SmartWings
Minimum airplane distance: 12.165525
Pairs found: 1
LOT - SmartWings
```

```
Plane coordinates:
-1000000,0: LOT
1000000,0: KLM
5000000,0: AirFrance
Minimum airplane distance: 2000000.000000
Pairs found: 1
LOT - KLM
```

```
Plane coordinates:
10,10: AirFrance
10,10: Lufthansa
20, 20: Ryanair
20,20: Wizz
20,20: Qantas
10,10: LOT
Minimum airplane distance: 0.000000
Pairs found: 6
AirFrance - Lufthansa
AirFrance - LOT
Lufthansa - LOT
Ryanair - Wizz
Ryanair - Qantas
Wizz - Qantas
```

```
Plane coordinates:
3,abc: CSA
Invalid input.
```

```
Plane coordinates:
0,0: LOT
5,8 KLM
Invalid input.
```

Notes:

- The sample runs above list both the output of your program (boldface font) and user input (regular font). The bold/regular formatting is included here, in the problem statement page, to increase readability of the listing. Your

program must output the text without any additional markup.
- Do not forget the newline (\n) after the last output line.
- The representation of the airplanes must be allocated dynamically.
- The size of the input is not known. The program shall start with a small array and increase the allocated memory size as the input is read. Function `realloc` is designed for this purpose.
- Do not mix C allocation functions (`malloc`, `realloc`, `free`, ...) and C++ allocation operators (`new`, `delete`). It is not recommended, moreover, your program may fail tests with memory debugger.
- Do not use C++ STL library (`list`, `vector`, ...). The purpose of this homework is to drill dynamic memory allocation. STL library will be used in PA1, it is forbidden in PA1. If you use STL, your program will not compile.
- The memory access and memory allocation is checked in one of the tests. The test checks whether all dynamically allocated memory was freed, or not. If your program fails to free all allocated memory, there will be a penalty.
- The printed distance is a floating point number. The testing environment permits a small difference when comparing these numbers.
- The program from this homework may be used for code review. However, only programs that pass all mandatory and optional tests with 100% results are eligible (i.e., bonus test is not required for code review).

Sample data: **Download**

Submit: Choose File **Submit**

Reference