# South African Computer Olympiad
# Camp 2, 2009
# Day 1


# Overview

| **Author(s)** | **Ben Steenhuisen and Richard Baxter** | **Max Rabkin (from ACM ICPC World Finals 2009)** | **Keegan Carruthers-Smith** | **Timothy Stranex** |
|---|---|---|---|---|
| **Problem** | **hotel** | **schedule** | **security** | **borg** |
| Source | hotel.c<br>hotel.cpp | schedule.c<br>schedule.cpp | security.c<br>security.cpp | borg.c<br>borg.cpp |
| Input file | stdin | stdin | stdin | stdin |
| Output file | stdout | stdout | stdout | stdout |
| Time limit | 1 second | 12 seconds | 1 second | 1 second |
| Number of tests | 10 | 10 | 10 | 10 |
| Points per test | 10 | 10 | 10 | 10 |
| Detailed feedback | Yes | No | No | No |
| **Total points** | **100** | **100** | **100** | **100** |

The maximum total score is 400 points.

http://olympiad.cs.uct.ac.za/contest.html

# Hotel

Ben Steenhuisen and Richard Baxter

## Introduction

The hotel inspectors have returned! This time they are checking whether too many guests have stayed at Fawlty Towers recently. Help Basil prove that Fawlty Towers has not been overcrowded by finding the maximum number of guests that have stayed at the hotel on any given day.

## Task

Basil has only recorded the check-in and check-out days for each guest. He never kept daily totals of the number of guests in his hotel. He needs to know the number of people that were in his hotel when it was at its fullest. Guests are considered at the hotel on the day of arrival and up until **and including** the day of departure.

The days are numbered from 0 onwards $(0, 1, 2, \ldots)$ and there were no guests in the hotel at the beginning of day 0. Manuel is at reception and he can only handle one departure or arrival per day, so there wil be at most one check-in or check-out on any day.

## Example

The hotel has 4 guests in its records. Below are the check-in and check out dates:

| Guest | Check-in day | Check-out day |
|---|---|---|
| 1 | 2 | 10 |
| 2 | 7 | 23 |
| 3 | 9 | 11 |
| 4 | 3 | 5 |

According to the above arrival and departure times the hotel was at its fullest on day 9, as there were three guests in the hotel on that day.

## Input (stdin)

The first line contains a single integer $N$: the number of guests. The next $N$ lines each contain two space-separated integers: the arrival day followed by the departure day of a guest.

### Sample input

```
4
2 10
7 23
9 11
3 5
```

Output a single integer: the maximum number of guests that have stayed in the hotel on any day.

## Output (stdout)

### Sample output

```
3
```

## Constraints

- $1 \leq N \leq 50\,000$

Additionally, in 50% of the test cases:

- $N \leq 5\,000$

## Time limit

1 second.

## Detailed feedback

Detailed feedback is enabled for this problem.

## Scoring

A correct solution will score 100%, while an incorrect solution will score 0%.

# Airport Scheduling

Max Rabkin (from ACM ICPC World Finals 2009)

## Introduction

Tired of endless lost baggage at Heathrow, and fearful of accidents, Bruce has decided to open his own airport, guided by the principles of maximum safety and extreme accuracy.

Air-traffic controllers try to maximize the time between planes landing on a runway for safety. Airlines, on the other hand, want their planes to land on time. Help them arrange a safe landing schedule that allows all the planes to land at the desired time.

## Task

Given a list of $N$ planes with their earliest and latest possible landing times, find a time for each plane to land such that the minimum time between successive landings is maximized.

Landing times of all planes must be a whole number of femtoseconds (when Bruce wants extreme accuracy, he doesn't mess about).

## Example

Plane 1 can land any time from 0 to 5, plane 2 can land between 5 and 6, and plane 3 must land between 6 and 7 (times are in femtoseconds past midnight).

The best schedule is for plane 1 to land at 0, plane 2 at 5 and plane 3 at 7; the minimum time between landings in this schedule is 2 femtoseconds.

## Input (stdin)

The first line of input contains a single integer, $N$: the number of planes. The next $N$ lines contain two space-separated integers $E_i$ and $L_i$: the earliest and latest possible landing time of the $i$th plane, in femtoseconds past midnight.

### Sample input

```
3
0 5
5 6
6 7
```

## Output (stdout)

Output a single integer: the largest possible minimum time between landings.

### Sample output

```
2
```

## Constraints

- $1 \leq N \leq 10$
- $0 \leq E_i \leq L_i \leq 10^{18}$

Additionally, in 20% of the test cases:

- $0 \leq E_i \leq L_i \leq 10^3$

If your program can only handle times up to $10^3$ femtoseconds, Bruce will only be able to open his airport for one picosecond per day, so you can see why you'll only get 20%.

## Time limit

12 seconds.

## Scoring

A correct solution will score 100% while an incorrect solution will score 0%.

# Security Guards

Keegan Carruthers-Smith

## Introduction

Fred the Manic Storekeeper has decided that he is a VIP. As such, there are now security guards guarding his store. Being a manic storekeeper, he has hired guards which are very pedantic about how they will be scheduled to guard the store.

The guard schedule is broken down into $M$ shifts. Each shift has two guards assigned to it. A guard will *always* work two shifts *consecutively* (and never more than two shifts consecutively). The guards also list who they will do a shift with. For any schedule, a guard only wants to work with another guard once.

If a guard $i$ wants to do a shift with guard $j$, it is assumed that guard $j$ also wants to do a shift with guard $i$. Note that the same schedule is repeated consecutively, so the rules above must be satisfied across schedules.

## Task

Your task is to find a schedule to suite the guards' fussy needs. This may not always be possible, and if so a schedule length of 0 should be output.

## Example

Say the following guard pairs are willing to work with each other:

$$(0,1),(0,2),(1,2),(2,3),(2,4),(3,4)$$

So we have $M = 6$ shifts to schedule. The following schedule is a valid schedule:

$$(0,1) \to (1,2) \to (2,3) \to (3,4) \to (2,4) \to (0,2)$$

Note that schedules get repeated, so we need the end of the schedule and beginning to match up. i.e. $(2,0) \to (0,1)$ happens in this schedule, which is why 0 actually works two shifts in the beginning.

$(0,1) \to (1,2) \to (0,1)$ is an invalid schedule since:

- Only 3 shifts where given, instead of 6
- $(0,1)$ is repeated
- 1 works 3 shifts consecutively. (instead of 2)
- 2 works 1 shift. (instead of 2)

## Input (stdin)

The first line of input contains two space-separated integers, $N$ and $M$. $N$ is the number of guards and $M$ is the number of guard pairs. The next $M$ lines contain two space-separated integers, $i$ and $j$. $i, j$ represent a pair of guard willing to work together.

### Sample input

```
5 6
0 1
0 2
1 2
2 3
2 4
3 4
```

## Output (stdout)

The first line of output contains $S$, the number of shifts in the schedule. $S = M$ if a schedule was successfully created, or $S = 0$ if no such schedule could be created. The next $S$ lines each contain a single integer $i$. $i$ is the number of the guard leaving his shift. Refer to the example and sample output for more information.

### Sample output

```
6
0
2
4
3
2
1
```

## Constraints

- $1 \leq N \leq 1\,000$
- $N \leq M \leq 10\,000$
- $0 \leq i, j < N$
- $S \in \{M, 0\}$
- Every guard will appear in at least one guard pair.

Additionally, in 30% of the test cases:

- $1 \leq N \leq 10$
- $N \leq M \leq 10$

# South African Computer Olympiad
# Camp 2, 2009
# Day 1

## Time limit

1 second.

## Scoring

A correct solution will score 100% while an incorrect solution will score 0%.

# The Borg Collective

Timothy Stranex

## Introduction

The Borg have finally succeeded in assimilating all species in the universe. However, this has left their starships scattered across the universe. They must now join together to complete the collective.

## Task

There are $N$ starships with the $i$th located at position $(x_i, y_i)$. They will regroup in several steps. In each step, the two closest ships cruise toward each other until they meet. Once they meet, they merge together to form a single ship. This process is repeated until only one ship remains.

The ships meet at the midpoint between them. So if ship $i$ and ship $j$ were to merge, the new ship would be located at

$$\left(\frac{x_i + x_j}{2}, \frac{y_i + y_j}{2}\right).$$

The original ships are numbered from 1 to $N$. The newly merged ships are numbered $N + 1$, $N + 2$, $N + 3$ and so on.

The distance between ship $i$ and ship $j$ is given by

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

## Example

Suppose there are $N = 3$ ships located at $(0, 1)$, $(1, 1)$ and $(2, 5)$. In the first step ship 1 and 2 merge to create ship 4 at $(0.5, 1)$. In the second step, ship 3 and 4 merge to create ship 5 at $(1.25, 3)$. This is now the only ship remaining so the process is complete.

## Input (stdin)

The first line of the input contains a single integer $N$: the initial number of ships. The next $N$ lines each contain two space-separated integers: $x_i$ and $y_i$, the positions of the ships.

### Sample input

```
3
0 1
1 1
2 5
```

## Output (stdout)

The output consists of $N - 1$ lines, one for each merging step. Each line contains two space-separated integers, the numbers of the two ships that merged during the step.

### Sample output

```
1 2
3 4
```

## Constraints

- $1 \leq N \leq 2\,000$
- $0 \leq x_i, y_i < 1\,000\,000$

Additionally, in 50% of the test cases:

- $1 \leq N \leq 100$

## Time limit

1 second.

## Scoring

A correct solution will score 100% while an incorrect solution will score 0%. There may be several correct solutions. Any correct solution is acceptable.