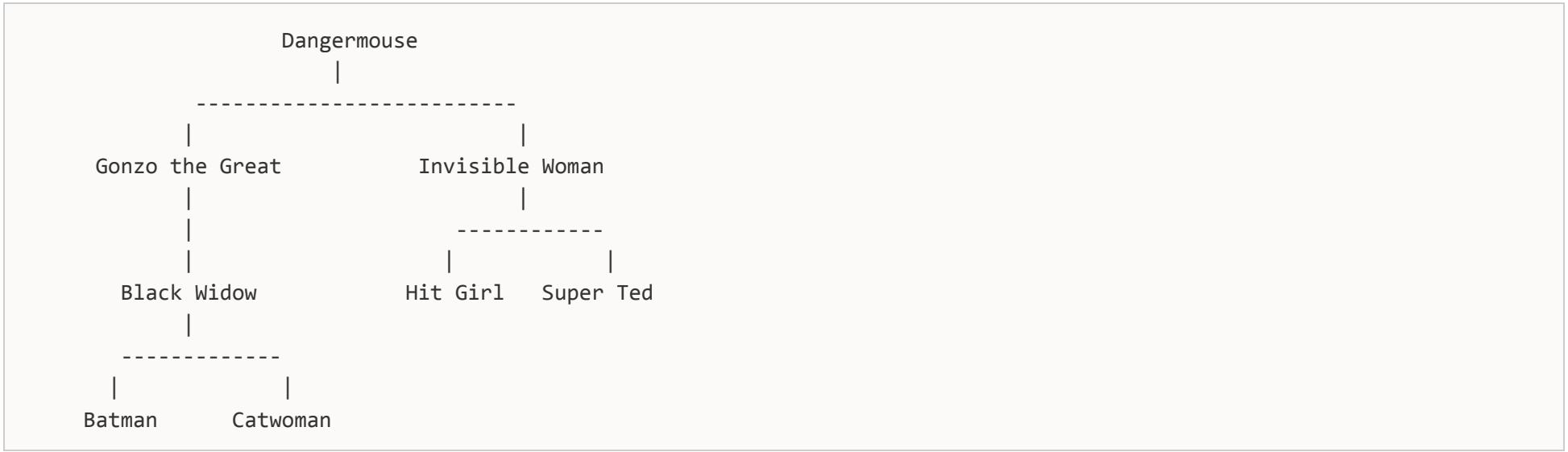
Organisation Chart Traversal

You work for Superheroes Inc, an old-fashioned company which has very strict rules about only passing information up and down the management chain.

Your job is to write a program which, given details of the organisational chart and the names of two people, prints out the names of all the people in the chain between them.

For example, here's a simple organisational chart:



A list of employees will be provided in a text file, one employee per line, in the following format (including the header line). This example corresponds to the diagram above:

```
| Employee ID | Name            | Manager ID |
| 1           | Dangermouse     |            |
| 2           | Gonzo the Great | 1          |
| 3           | Invisible Woman | 1          |
| 6           | Black Widow     | 2          |
| 12          | Hit Girl        | 3          |
| 15          | Super Ted       | 3          |
| 16          | Batman          | 6          |
| 17          | Catwoman        | 6          |
```

There is no guarantee in which order the employees will be listed. The employee ID will always be an integer, as will the manager ID, except for the person at the top of the hierarchy, where it will be blank. There may be gaps in the sequence (where people have left the company).

Your program will receive three separate command line arguments. The first is the path to the filename containing the employee data, and the other two are employee names. It should print the shortest path of communication between them that follows the hierarchy. So if these names were “Batman” and “Super Ted”, the program will be run like this:

```
MyProgram superheroes.txt "Batman" "Super Ted"
```

and it should give the following output:

```
Batman (16) -> Black Widow (6) -> Gonzo the Great (2) -> Dangermouse (1) <- Invisible Woman (3) <- Super Ted (15)
```

The numbers in brackets are the employee IDs. If the names were “Batman” and “Catwoman”, however, it would give:

```
Batman (16) -> Black Widow (6) <- Catwoman (17)
```

There is no guarantee that names are unique – your program should print at least one path between each pair of people with the same name, but you don't have to show, for example, the path from “Minion (1)” to “Minion (2)” **and** the path from “Minion (2)” to “Minion (1)”.

All output lines must be in the in the format given in the examples (including the ID), and showing the names as found in the input file.

When comparing names, the case of letters is not significant, and neither are leading or trailing spaces or runs of multiple spaces. So:

```
"Gonzo the Great"
"gonzo the GREAT"
" Gonzo   the Great "
```

are all considered to be equivalent, but:

```
"Gon Zot Heg Reat"
```

is not. Note that the quotes in these examples are only to show you where the spaces are, your program does not have to strip quotes.