ECE 36800 Assignment #1

```
Original Due: 1:00 PM, Tuesday,  September 9
    Extended: 1:00 PM, Thursday, September 11
```

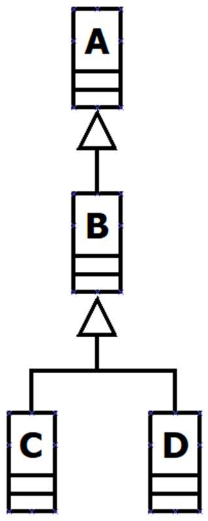
You must complete this assignment on your own. You may not work with anyone else in the class or with anyone outside the class. Copying solutions from the internet is not allowed. All code you submit must be your own work. You may not share your assignment with anyone or make it publicly available.

Description

Brute force is a straightforward approach to solving a problem, typically based directly on the problem statement and the definitions of the concepts involved. For this warm-up assignment, you will write a program that describes the class hierarchy (i.e., inheritance relationships between parent and child classes) in the Java programming language. The semantics of inheritance in Java are not significant for this assignment, except to the extent that a parent class can have multiple child classes, but each child class can have only one parent class. You will write and submit code only written in the **C** programming language. Your program will take as a command-line argument the name of a text file containing a sequence of parent-child class names and print the class hierarchy to standard output in a specified format.

Input Format

Consider the class hierarchy shown below, where class `A` is the parent of class `B`, and `B` is the parent class of both `C` and `D`. For simplicity, each class name will be a single uppercase English letter (e.g., `A`, `B`, `C`).



Each line of input represents a parent-child class pair, with the parent class name first. For example, `AB` means that `A` is the parent class of `B`. Below is an input example that describes the inheritance relationship above:

```
AB
BC
BD
```

The order of the input lines does not affect the output, as long as the inheritance (parent-child class) relationships remain the same. All six combinations below will result in the same output:

Input #1	Input #2	Input #3	Input #4	Input #5	Input #6
AB BC BD	AB BD BC	BC AB BD	BC BD AB	BD AB BC	BD BC AB

Output Format

Given the example input above in a file named `input.txt`, an sample run of the program is shown below:

```
$ ./a1 input.txt
A
B
CD
# #
```

Output rules:

- Each line displays the names of all siblings (i.e., children of the same parent).
- Siblings must be listed in alphabetical order (e.g., `CD` is correct, `DC` is not).
- Insert a line feed between parent-child levels in the hierarchy.
- Use a `#` symbol to represent a parent with no children.
- Insert a space between groups of siblings to indicate that they have different parents.
- There must be a blank line at the end of the output.

More Examples

Examples	Input	Output
#2	AB BC BD AE EF	A BE CD F # # #
#3	AB BC BD EF	A B CD # # E F #
#4	AB BC BD AE	A BE CD # # #
#5	AB AC AD CE	A BCD # E # #
#6	AB BC AC	INVALID
#7	AB BA	INVALID

Example #2. There is a space between CD and F because they have different parents B and E, respectively. Also, since C, D and F have no children, their children are represented by # symbols, separated by spaces.

Example #3. This input contains multiple class hierarchies. Each distinct hierarchy is separated by a blank line in the output. The order of the hierarchies follows the alphabetical order of the root (top) class names. In the example, the hierarchy of A is displayed before the hierarchy of E.

Example #4. The # symbol cannot represent a parent with children. The last line displays only the children of C and D, which are represented as # symbols.

Example #5. Another example. The last line displays only the children of E.

Example #6. C has two parents, B and A. Java does not support multiple inheritance for classes—each child class can have only one parent class. Your program should simply print INVALID for invalid input.

Example #7. The input is invalid due to circular inheritance—A is the parent of B, and B is the parent of A.

Grading

This assignment will be graded based on both the correctness and memory safety of your implementation. Each test case will be worth the same amount of points, and will be all-or-nothing (i.e. you either have the correct output or you don't). Note that even minor differences in output will cause you to fail grading test cases. Use a `diff` tool to ensure your program produces the exact expected output. A sample test case is provided for your use.

You may submit to Gradescope as often as you'd like before the deadline. Only your active submission (by default the most recent) will be counted. While the score given to you by the autograder is likely a good indicator of your final grade on the assignment, we reserve the right to add additional test cases after the submission deadline.

Submission

Submit any source/header files with your implementation, as well as a Makefile that builds a target called `a1`, to Gradescope. Note that to receive points, your submission must work on eceprog.