Part 1:

I like to start from the most challenging parts, so I'm going to figure out this problem, starting from the logic to find the dependencies

We are going to save the libraries and its dependencies using a literal object call dependenciesTable

Given an str, first we need to separate them in lines, so:

```
lines: List = str.split("\n")
```

Every line should contain the following pattern

"lw depends on lwt"

If some line doesn't follow the pattern it will be discarded

Now with all the lines validated, I'm going to figure out how to store them.

!Notice that we can do this right away after verifing the line to avoid traversing the list again

So we look up for the library and their dependencies

the words depends on (with spaces) could be used as a separator

```
match = line.split(" depends on ")
library = match[0]
dependencies_str = match[1]
dependencies = dependencies_str.split(" ") ← This should be a set to avoid duplication
```

So we add library as a key and the dependencies as its value

```
dependenciesTable[library] = dependencies
```

with this done we can follow the next step

Something important to notice before start!

For example if we have the following string

input	output
A → B C	A → B C E F G H
B → C E	B → C E F GH
C → G	C → G
D → A F	D → A B C E F G H
E → F	E → F H
F → H	F → H

- Notice that their values are ordered alphabetically
- this is not exactly a matter of concern. But there's some incertity of how other people would make their own test

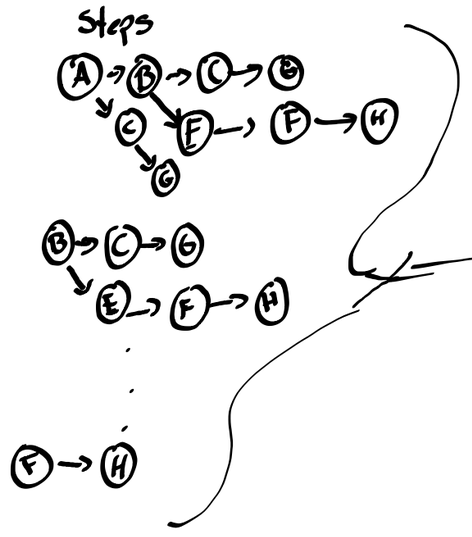
  Just for this I'll make the logic for ordering using lexicon
- Also we need to avoid circular dependency

Part 2:

Now I'm going to figure at how to find the nested solutions, and here is when using a literal object is really convenient

dependencies table

```
"A": ["B", "C"]
"B": ["C", "E"]
"C": ["G"]
"D": ["A", "F"]
"E": ["F"]
"F": ["H"]
```



It seems that we can use iteration and recursion

```
for library in dependencyTable.keys
  newPath = new Set()
  findAllLibraries(libraryDependencyTable, newPath, library)
  dependenciesTable[library] = newPath
```

```
findAllLibraries(key, obj, path, origin)
  if key not in obj:
    return ← recursive case
  for k in obj[key]
    if key == origin || path.has(k) continue; ← to avoid circular regression
    findAllLibraries(k, obj, path, origin)
    path.add(k)
```

Then after that we should order the values alphabetically

```
sortAlphabetically(obj)
  for library in obj{
    array = Array.from(obj[library])
    array.sort()
    obj[library] = dependencies Array
```

And that's it!

Now I'm going to skip the logic of reading from a file and print the output, because it's worthless and can be done easily