# Config Challenge

## Marking Criteria

When we read your submission we'll consider the following:

- **Approach**: How are you solving this problem? What alternative approaches did you consider? Does your submission meet the requirements of this brief?

- **Code structure and quality**: Would it be easy for others to maintain your code? Is it extensible? Did you apply thoughtful *separation of concerns* and *domain modelling* – have you produced a *well-designed code architecture*?

- **Documentation**: Can you explain and justify the approach you've taken? What were the key decisions and assumptions you made and why? Can you provide appropriate documentation so that others on your team can work with your code?

Guidelines:

- Please use whatever language you feel most comfortable with for this task. Feel free to adapt the signature of the `loadConfig` method appropriately for your language.

- Please include a README with information on how to execute and test your code.

- Please do not spend more than 3 hours on this challenge. Rather than spending more time, please leave notes in the README as to what you would do if you had more time.

- Please implement the parsing logic yourself. It might be good software engineering practice to use a library, but in this case it makes it harder to evaluate your submission.

- Feel free to use Google, Stack Overflow or other sources of information as much as you like, but do write the code yourself.

# Challenge

Every large software project has its share of configuration files to control settings, execution and more. You are given a config file format that will appear as follows:

```
; This is the config file format your code
should accept.

[common] ; this denotes the start of a group
called common
basic_size_limit = 26214400
student_size_limit = 52428800
paid_users_size_limit = 2147483648
path = /srv/var/tmp/
path<itscript> = /srv/tmp/

[ftp]
name = "hello there, ftp uploading"
path = /tmp/
path<production> = /srv/var/tmp/
path<staging> = /srv/uploads/
path<ubuntu> = /etc/var/uploads
enabled = no
; This is a comment

[http]
name = "http uploading"
path = /tmp/
path<production> = /srv/var/tmp/
path<staging> = /srv/uploads/; This is another
comment
params = array,of,values
```

## Keys

| | |
|---|---|
| `[group]` | This denotes the start of a group of related config options. |
| `setting = value` | A setting and associated value. |
| `setting<override> = overridden_value` | A setting and value if the given override is enabled. If multiple overrides are defined, the one defined last will have priority. |

# Assignment

Your task is to write a function that parses the input format defined above and returns an object:
`loadConfig(String file_path, List overrides)`

An example call would look like:
`config = loadConfig("/srv/settings.conf", ["ubuntu", "production"])`

The object returned should be queryable like this. ***You are expected to adapt this interface to your code design and the conventions of the language you are using***:

```
config.get("common").get("paid_users_size_limit")
# should be an int 2147483648

config.get("ftp").get("name")
# should be a string "hello there, ftp uploading"

config.get("ftp").get("lastname")
# should be a suitable empty value as it doesn't exist


config.get("http").get("params")
# should be an array ["array", "of", "values"]

config.get("ftp").get("enabled")
# should be a boolean false
# permitted bool values are "yes", "no", "true",
"false", 1, 0

config.get("ftp").get("path")
# should be a string "/etc/var/uploads"

config.get("ftp")
# should be a mapping { name => "hello there, ftp
uploading", path => "/etc/var/uploads", enabled => False
}
```

# Design Considerations

When implementing your solution, please consider the following:

- `loadConfig()` will be called at boot time, and thus should be efficient with time. Config

files can get quite lengthy - there can be an arbitrary number of groups and number of
settings within each group.

- The `Config` object will be queried throughout the program's execution, so each query should be fast as well.

- Certain queries will be made frequently (thousands of times), others pretty rarely.

- If the config file is not well-formed, it is acceptable to print an error and exit from within

  `loadConfig()`. Once the object is returned, however, it is not permissible to exit or crash no matter what the query is. Returning an empty value is acceptable, however.