Config Challenge

Using Scala and a parser combinator library of your choice, please spend 3 hours on:

Every large software project has its share of configuration files to control settings, execution, etc. Let's contemplate a config file format that looks a lot like standard PHP .ini files, but with a few tweaks.

A config file will appear as follows:

```
[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
```

Where "[group]" denotes the start of a group of related config options, `setting = value` denotes a

standard setting name and associated default value, and `setting<override> = value2` denotes the value for the setting if the given override is enabled. If multiple enabled overrides are defined on a setting, the one defined last will have priority.

Assignment

Your task is to write a Scala function: `def loadConfig(filePath: String, overrides: List[(String, String)])` that parses this format and returns an object that can be queried as follows. Example method call with overrides: `val config = loadConfig("/srv/settings.conf", List("ubuntu" -> "production"))`

```
>>> config("common.paid_users_size_limit")

# returns 2147483648

> config("ftp.name")

# returns "hello there, ftp uploading"

>>> config("http.params")

# returns ["array", "of", "values"]

> config("ftp.lastname")

# returns None

> config("ftp.enabled")

# returns false (permitted bool values are "yes", "no", "true", "false", 1, 0)

> config("ftp['path']") # returns "/etc/var/uploads"

> config("ftp")

# returns a map: # { # 'name' => "hello there, ftp uploading", # 'path' =>
"/etc/var/uploads", # 'enabled' => False # }
```

We expect a full Scala solution with the appropiate amount of tests included as well.

Design Considerations

- `loadConfig()` will be called at boot time, and thus should be as fast as possible. Conf files can get quite lengthy - there can be an arbitrary number of groups and number of settings within each group.
- `config` will be queried throughout the program's execution, so each query should be very fast as well.
- Certain queries will be made very often (thousands of times), others pretty rarely.
- If the conf 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 `None` is acceptable, however.
- Extra points awarded for declarative config access. `config.ftp.path` would validly compile and return the same as `config("ftp.path")`.