Check out our coding help page to ensure you get a good score.

Getting started

1. Getting the output right is important, but more important is clean code and how well designed your code is. You should **absolutely** see our Help page post on what we look for in your code, and how to get started with the coding challenge.

2. See our evaluation parameters here and the badges to earn here.

3. We expect a command line app. So no web apps will be considered for evaluation. You don’t need data stores either.

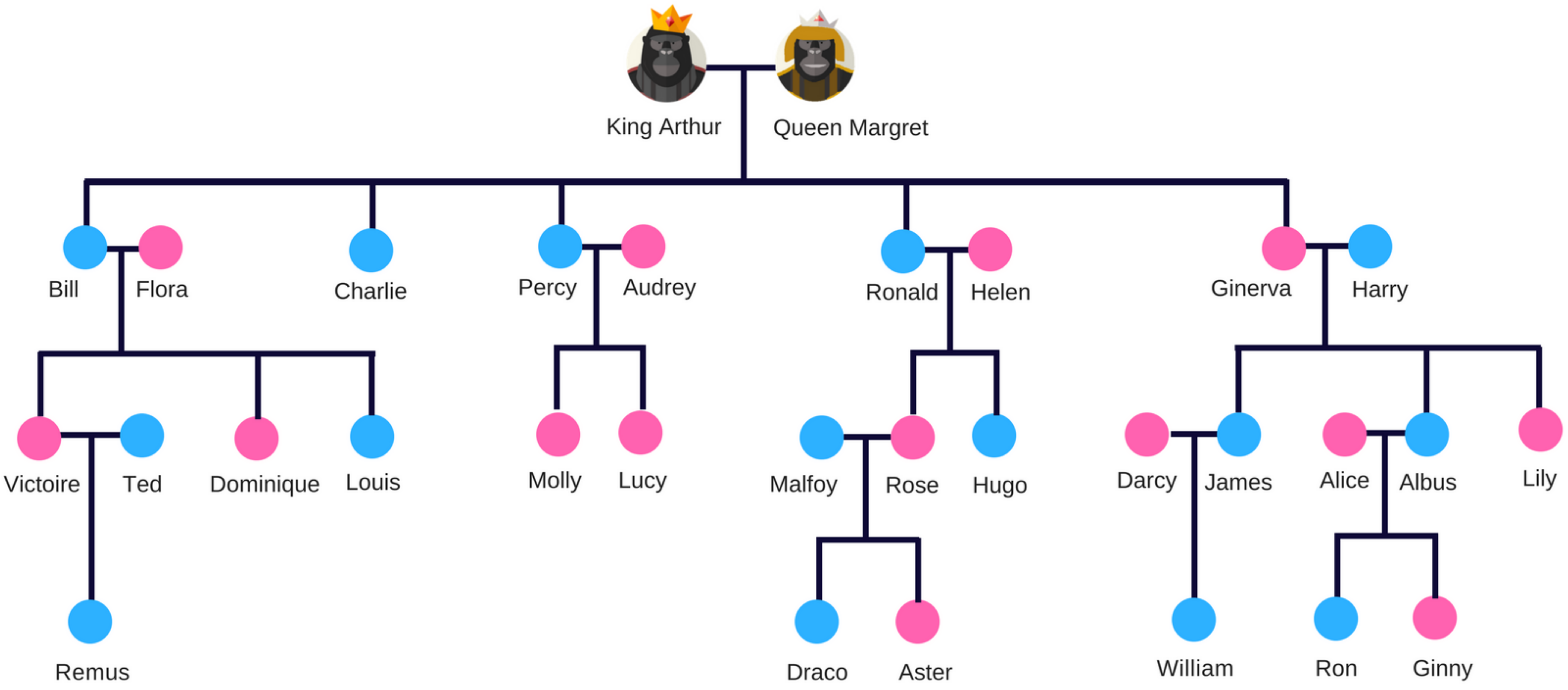
Problem Context

Our story is set in the planet of Lengaburu......in the distant, distant galaxy of Tara B. And our protagonists are King Arthur, Queen Margret & their family.

King Arthur is the emperor of Lengaburu and has been ruling the planet for the last 350 years (they have long lives in Lengaburu, you see!). Let's write some code to get to know the family.

This coding problem is for backend and fullstack developers.

Family Tree



The Arthur Family Tree

● Male ● Female

Meet The Family

Write code to model out the King Arthur family tree so that:

- Given a ‘name’ and a ‘relationship’, you should output the people corresponding to the relationship in the order in which they were added to the family tree. Assume the names of the family members are unique.

- You should be able to add a child to any family in the tree through the mother.

Simple, right? But remember.. our evaluation is based not only on getting the right output, but on how you've written your code.

Relationships To Handle

There are many relations that could exist but at a minimum, your code needs to handle these relationships.

Relationships	Paternal-Uncle	Maternal-Uncle	Paternal-Aunt	Maternal-Aunt	Sister-In-Law	Brother-In-Law	Son	Daughter	Siblings
Definition	Father’s brothers	Mother’s brothers	Father’s sisters	Mother’s sisters	Spouse’s sisters, Wives of siblings	Spouse’s brothers, Husbands of siblings			

Sample Input/Output

Do initialise the existing family tree on program start. **Your program should take the location to the test file as parameter.** Input needs to be read from a text file, and output should be printed to the console. The test file will contain only commands to modify or verify the family tree.

Input format to add a child:

```
ADD_CHILD  "Mother's-Name"   "Child's-Name"   "Gender"
```

Input format to find the people belonging to a relationship:

```
GET_RELATIONSHIP  "Name"   "Relationship"
```

Output format on finding the relationship:

```
"Name 1"   "Name 2"… "Name N"
```

Example test file:

```
ADD_CHILD Flora  Minerva  Female
GET_RELATIONSHIP Remus  Maternal-Aunt
GET_RELATIONSHIP Minerva Siblings
```

Output on finding the relationship:

```
CHILD_ADDITION_SUCCEEDED
Dominique Minerva
Victoire Dominique Louis
```

More sample input output scenarios.

Please stick to the Sample input output format as shown. This is very important as we are automating the correctness of the solution to give you a faster evaluation. You can find some sample input output files here.

Sample 1

```
ADD_CHILD   Luna  Lola Female
GET_RELATIONSHIP Luna Maternal-Aunt
```

Sample 1 Output:

```
PERSON_NOT_FOUND
PERSON_NOT_FOUND
```

Luna does not exist in the family tree

Sample 2

```
ADD_CHILD   Ted  Bella Female
GET_RELATIONSHIP Remus Siblings
```

Sample 2 Output:

```
CHILD_ADDITION_FAILED
NONE
```

Ted is male, hence child addition failed

Sample 3

```
GET_RELATIONSHIP Lily Sister-In-Law
```

Sample 3 Output:

```
  Darcy Alice
```

C# - Instructions to Build & Execute

We support **.NET Core 2.2 & 3.1** for C# applications. The only requirement here is you should add an **AssemblyName** entry with the value **geektrust** in your **.csproj** file. This will ensure that the **dll** file created will be named **'geektrust.dll'**.

We then build and execute the solution by the following commands. Read more.

```
dotnet build -o geektrust
dotnet geektrust/geektrust.dll <absolute_path_to_input_file>
```

Go - Instructions to Build & Execute

Management of dependencies needs to be done via **Go Modules**. Do provide the **go.mod** file if you are using it. We use **go tool** for building Go applications. All your code should reside inside a package named **geektrust** under your Go workspace, which is typically the **GOPATH**. The name of the package should be geektrust and **NOT** be anything else.

Create your geektrust package path with the command.

```
mkdir $GOPATH/src/geektrust
```

Your directory structure should then look like this.

```
bin/
  geektrust                  # command executable
src/
  geektrust                  # main package
    main.go                  # start program file
    file_1.go                # another file required
    subpackage1              # a sub package you may write
      subpackage1.go         # a file under that sub package
```

We build and execute the the solution by using the following command from the directory **$GOPATH/src/geektrust**. The executable **geektrust** will be generated in the directory **$GOPATH/src/geektrust** besides the **main.go** file. Read more.

```
go build .
./geektrust <absolute_path_to_input_file>
```

Java - Instructions to Build & Execute

Solution with build file

If you are writing code in Java with a build system, you have to use either **maven** or **gradle** with the respective build files. Please download them from the links below.

pom.xml
build.gradle

Edit the build file to set your '{your.qualified.name.of.main.class}' and add your dependencies if any. Ensure the generated executable is named **'geektrust.jar'.**

Post build we then execute the solution by the following command. Read more.

```
java -jar geektrust.jar <path_to_input_file>
```

Solution without build file

For a solution without build system, we want you to name your **Main** class as **Geektrust.java**. This is the file that will contain your main method. We build and run the solution by using the following commands. Read more.

```
javac <path_of_package>/Geektrust.java
java <package>.Geektrust <absolute_path_to_input_file>
```

Node - Instructions to Build & Execute

We execute NodeJS projects using Yarn/NPM execution environment. For Standalone Javascript we use Node execution environment. In all cases your **main file** should be named as **geektrust.js**.

Solution with build file

We support both **NPM** and **Yarn** build systems. Your project should have the **package.json** with all your dependencies. In that file make sure you have an entry for the start script which points to the execution of geektrust.js.

We then execute the solution by the following command. Read more.

NPM

```
npm install --silent
npm start --silent <absolute_path_to_input_file>
```

Yarn

```
yarn install --silent
yarn run --silent start <absolute_path_to_input_file>
```

Solution without build file

For a solution without build system, we want you to name your **main** file as **geektrust.js**. This is the file that will contain your main method. We then execute the solution by the following command. Read more.

```
node geektrust.js
```

Python - Instructions to Build & Execute

Solution with build file

We support only **Pip** build system. You can add your dependent packages and versions in requirements.txt.

The **Main** file should be named as **geektrust.py**. This is the file that will contain your main method.

We then execute the solution by the following command. Read more.

```
pip install -r requirements.txt
python -m geektrust <absolute_path_to_input_file>
```

Solution without build file

For a solution without build system, we want you to name your **Main** file as **geektrust.py**. This is the file that will contain your main method. We then execute the solution by the following command. Read more.

```
python -m geektrust <absolute_path_to_input_file>
```

Ruby - Instructions to Build & Execute

Solution with build file

We support only **Rake** build system. The rake file should import the main ruby file, which is the starting point of your application and call the **main** method within the **default task**. Also create a **Gemfile** to add your dependencies, if any. The main file should read the file path from the **ARGV** variable and then execute the program.

The solution will be run using the following commands. Read more.

```
bundle install
rake default <absolute_path_to_input_file>
```

Solution without build file

For a solution without build system, we want you to name your **Main** file as **geektrust.rb**. This is the file that will contain your main method.

The solution will be run using the following commands. Read more.

```
ruby -W0 geektrust.rb <absolute_path_to_input_file>
```

Instructions to Build & Execute

If you are writing code in a language other than, Java, C#, Python, NodeJs, Ruby, Go; please make sure:

- Your application should be a **command line application**.
- Your main file to execute is named as **geektrust.<file_ext>**.
- It takes in a command line argument which is the location of the text file containing the commands that needs to be executed by your program.

  For e.g, if you re solving in **PHP**, and your input file is ***/tmp/input1.txt*** then the command for executing your code should be:

```
php geektrust.php /tmp/input1.txt
```

- After processing, it should print only the output related to each command in the file.

  For e.g , if the input file passed in has commands like :

```
INPUT_COMMAND input1
INPUT_COMMAND input2
```

then your solution should print only :

```
Output for input1
Output for input2
```

Supported Language & Versions

Code submissions are run against a Linux virtualized instance.

Supported language and versions are below:

Language	Supported versions	Supported Tools
C#	dotnet core 2.2, 3.1	dotnet
Go	1.12.x	Go build tool
Java	1.8, 1.11	maven, gradle
Node.js	8.16.x, 10.16.x, 12.6.x	npm, yarn
Python	3.7, 3.8	pip
Ruby	1.9.x, 2.2.x, 2.6.x	rake, bundler-rake

You can upload code in any version of Clojure, C++, Erlang, Groovy, Kotlin, PHP, Scala. We don't have automated tests for these languages yet. So your evaluation will take longer than the others.

Check list - submitting code

1. Please compress the file before upload. We accept .zip, .rar, .gz and .gzip
2. Name of the file should be the problem set name you are solving. For e.g. if you have solved Family problem, please name your file ‘Family.zip’.
3. Please upload only source files and do not include any libraries or executables or node_modules folder.
4. Usage of non-essential 3rd party libraries will affect your evaluation.
5. Add a **readme** with how to get your code working, and how to test your code.
6. Your solution will be downloaded & seen by companies you’re interested in. Hence we advise you to provide a solution that will work on any system without any code changes/manual setup.