Task 1: Time on Earth (30%)

Task Description

Write a program that converts a number of seconds into seconds, minutes, hours and days.

> We write all units in plural, regardless of the quantity (e.g. `0 days`, `1 seconds`).

Input and Output Examples

Example 1

User input & program output

```
TIME IN SECONDS
Duration in seconds:
121
DATE AND TIME
The date and time is 0 days 0 hours 2 minutes 1 seconds
```

User input

```
121
```

Program output

```
TIME IN SECONDS
Duration in seconds:
DATE AND TIME
The date and time is 0 days 0 hours 2 minutes 1 seconds
```

Example 2

User input & program output

```
TIME IN SECONDS
Duration in seconds:
3600
DATE AND TIME
The date and time is 0 days 1 hours 0 minutes 0 seconds
```

User input

```
3600
```

Program output

```
TIME IN SECONDS
Duration in seconds:
DATE AND TIME
The date and time is 0 days 1 hours 0 minutes 0 seconds
```

Example 3

User input & program output

```
TIME IN SECONDS
Duration in seconds:
1760812
DATE AND TIME
The date and time is 20 days 9 hours 6 minutes 52 seconds
```

User input

```
1760812
```

Program output

```
TIME IN SECONDS
Duration in seconds:
DATE AND TIME
The date and time is 20 days 9 hours 6 minutes 52 seconds
```

The files for this example are provided. You can:
- Run the command `python3 time_on_earth.py < 1_60_60_24_0_0_0_1760812.in > my_output.out` in the terminal and press Enter to write the output of your program (i.e., time_on_earth.py) to the file (i.e., my_output.out).
- Run the command `diff my_output.out 1_60_60_24_0_0_0_1760812.out` in the terminal and press Enter to compare the output of your program (i.e., my_output.out) to the correct output (i.e., 1_60_60_24_0_0_0_1760812.out).
You may not need this, so only use it if it helps you.

Task 2: Time on Trisolaris (50%)

This task is similar to Task 1, but the rate of conversion between units is not fixed.

Task Description

Write a program that converts a number of seconds into seconds, minutes, hours and days, where the conversion from one unit to the other is defined at the beginning of the program, rather than be fixed.

Input and Output Examples

In the examples below and the automated tests, there is a typo in the line "Number of hours in an day on Trisolaris:".
As some of you have already completed this task, it is too late to fix this typo, so please roll with it. Thanks!

Example 1

User input & program output

```
TIME CONFIGURATION ON TRISOLARIS
Number of seconds in a minute on Trisolaris:
60
Number of minutes in an hour on Trisolaris:
60
Number of hours in an day on Trisolaris:
24
TIME IN SECONDS
Duration in seconds:
121
DATE AND TIME ON TRISOLARIS
The date and time on Trisolaris is 0 days 0 hours 2 minutes 1 seconds
```

User input

```
60
60
24
121
```

Program output

```
TIME CONFIGURATION ON TRISOLARIS
Number of seconds in a minute on Trisolaris:
Number of minutes in an hour on Trisolaris:
Number of hours in an day on Trisolaris:
```

```
TIME IN SECONDS
Duration in seconds:
DATE AND TIME ON TRISOLARIS
The date and time on Trisolaris is 0 days 0 hours 2 minutes 1 seconds
```

Example 2

User input & program output

```
TIME CONFIGURATION ON TRISOLARIS
Number of seconds in a minute on Trisolaris:
10
Number of minutes in an hour on Trisolaris:
10
Number of hours in an day on Trisolaris:
10
TIME IN SECONDS
Duration in seconds:
1000
DATE AND TIME ON TRISOLARIS
The date and time on Trisolaris is 1 days 0 hours 0 minutes 0 seconds
```

User input

```
10
10
10
1000
```

Program output

```
TIME CONFIGURATION ON TRISOLARIS
Number of seconds in a minute on Trisolaris:
Number of minutes in an hour on Trisolaris:
Number of hours in an day on Trisolaris:
TIME IN SECONDS
Duration in seconds:
DATE AND TIME ON TRISOLARIS
The date and time on Trisolaris is 1 days 0 hours 0 minutes 0 seconds
```

Example 3

User input & program output

```
TIME CONFIGURATION ON TRISOLARIS
Number of seconds in a minute on Trisolaris:
100
Number of minutes in an hour on Trisolaris:
53
Number of hours in an day on Trisolaris:
77
```

```
TIME IN SECONDS
Duration in seconds:
14420823
DATE AND TIME ON TRISOLARIS
The date and time on Trisolaris is 35 days 25 hours 48 minutes 23 seconds
```

User input

```
100
53
77
14420823
```

Program output

```
TIME CONFIGURATION ON TRISOLARIS
Number of seconds in a minute on Trisolaris:
Number of minutes in an hour on Trisolaris:
Number of hours in an day on Trisolaris:
TIME IN SECONDS
Duration in seconds:
DATE AND TIME ON TRISOLARIS
The date and time on Trisolaris is 35 days 25 hours 48 minutes 23 seconds
```

The files for this example are provided. You can:
- Run the command `python3 time_on_trisolaris.py < 2_100_53_77_0_0_0_14420823.in > my_output.out` in the terminal and press Enter to write the output of your program to the file.
- Run the command `diff my_output.out 2_100_53_77_0_0_0_14420823.out` in the terminal and press Enter to compare the output of your program to the correct output.
You may not need this, so only use it if it helps you.

Task 3: Time addition on Trisolaris (20%)

This task uses the same unit conversion as Task 2, but this time we're adding times!

Task Description

Write a program that asks the user for the current time of day (in hours, minutes and seconds), then for a number of seconds to wait, starting from that time of the day. It then displays the date (in days, hours, minutes and seconds) corresponding to when the wait is over. Again, the conversion from one unit to the other is defined at the beginning of the program.

For example:

- The user specifies earth times (60, 60, and 24 for the first 3 inputs).
- The user specifies that the time of the day is 10:30:00 (10, 30, and 0 for the next 3 inputs).
- The user specifies to wait a duration of 3661 seconds (3661 for the last input).
- The program outputs the day and time it is after waiting. Since 3661 seconds correspond to 1 hour, 1 minute and 1 second, the program would output that the day is (still) day 0 and the time is 11:31:01.

Input and Output Examples

Example 1

User input & program output

User input

```
60
60
24
0
0
0
121
```

Program output

```
TIME CONFIGURATION ON TRISOLARIS
Number of seconds in a minute on Trisolaris:
60
Number of minutes in an hour on Trisolaris:
```

```
60
Number of hours in an day on Trisolaris:
24
CURRENT TIME OF DAY ON TRISOLARIS
Hour of day on Trisolaris:
0
Minute of day on Trisolaris:
0
Second of day on Trisolaris:
0
DURATION TO WAIT
Duration in seconds:
121
DATE AND TIME ON TRISOLARIS AFTER WAITING
The date and time on Trisolaris after waiting is 0 days 0 hours 2 minutes 1 seconds
```

Example 2

User input & program output

```
TIME CONFIGURATION ON TRISOLARIS
Number of seconds in a minute on Trisolaris:
10
Number of minutes in an hour on Trisolaris:
10
Number of hours in an day on Trisolaris:
10
CURRENT TIME OF DAY ON TRISOLARIS
Hour of day on Trisolaris:
9
Minute of day on Trisolaris:
9
Second of day on Trisolaris:
9
DURATION TO WAIT
Duration in seconds:
1
DATE AND TIME ON TRISOLARIS AFTER WAITING
The date and time on Trisolaris after waiting is 1 days 0 hours 0 minutes 0 seconds
```

User input

```
10
10
10
9
9
9
1
```

Program output

```
TIME CONFIGURATION ON TRISOLARIS
Number of seconds in a minute on Trisolaris:
```

```
Number of minutes in an hour on Trisolaris:
Number of hours in an day on Trisolaris:
CURRENT TIME OF DAY ON TRISOLARIS
Hour of day on Trisolaris:
Minute of day on Trisolaris:
Second of day on Trisolaris:
DURATION TO WAIT
Duration in seconds:
DATE AND TIME ON TRISOLARIS AFTER WAITING
The date and time on Trisolaris after waiting is 1 days 0 hours 0 minutes 0 seconds
```

Example 3

User input & program output

```
TIME CONFIGURATION ON TRISOLARIS
Number of seconds in a minute on Trisolaris:
48
Number of minutes in an hour on Trisolaris:
56
Number of hours in an day on Trisolaris:
48
CURRENT TIME OF DAY ON TRISOLARIS
Hour of day on Trisolaris:
40
Minute of day on Trisolaris:
32
Second of day on Trisolaris:
20
DURATION TO WAIT
Duration in seconds:
669678
DATE AND TIME ON TRISOLARIS AFTER WAITING
The date and time on Trisolaris after waiting is 6 days 1 hours 40 minutes 2
seconds
```

User input

```
48
56
48
40
32
20
669678
```

Program output

```
TIME CONFIGURATION ON TRISOLARIS
Number of seconds in a minute on Trisolaris:
Number of minutes in an hour on Trisolaris:
Number of hours in an day on Trisolaris:
CURRENT TIME OF DAY ON TRISOLARIS
Hour of day on Trisolaris:
```

```
Minute of day on Trisolaris:
Second of day on Trisolaris:
DURATION TO WAIT
Duration in seconds:
DATE AND TIME ON TRISOLARIS AFTER WAITING
The date and time on Trisolaris after waiting is 6 days 1 hours 40 minutes 2
seconds
```

The files for this example are provided. You can:
- Run the command `python3 time_addition.py < 3_48_56_48_40_32_20_669678.in > my_output.out` in the terminal and press Enter to write the output of your program to the file.
- Run the command `diff my_output.out 3_48_56_48_40_32_20_669678.out` in the terminal and press Enter to compare the output of your program to the correct output.
You may not need this, so only use it if it helps you.

Task 4: One secret no hints (30%)

Task Description

Write a program that asks the user to guess a secret number from among a predetermined set, stored as a python list `[3, 1, 7, 9, 10]`.

The program repeatedly ask the user for a guess until it is one of the secrets.

Input and Output Examples

Example 1

User input & program output

```
Input a guess:
10
Correct!
```

User input

```
10
```

Program output

```
Input a guess:
Correct!
```

Example 2

User input & program output

```
Input a guess:
2
Incorrect!
Input a guess:
1
Correct!
```

User input

```
2
1
```

Program output

```
Input a guess:
Incorrect!
Input a guess:
Correct!
```

Task 5: All secrets no hints (40%)

This task is similar to Task 4, but the user now needs to find all secrets.

Task Description

Write a program that asks the user to guess all secret numbers of a predetermined set, stored as a python list `[3, 1, 7, 9, 10]`.

Input and Output Examples

> You may temporarily change the list of secrets to find if that helps your debugging, but make sure to revert it before running automated tests, as they rely on the list being unchanged.

Example 1

User input & program output

```
Input a guess:
3
Correct!
Input a guess:
7
Correct!
Input a guess:
10
Correct!
Input a guess:
9
Correct!
Input a guess:
1
Correct!
You have guessed all the secrets!
```

User input

```
3
7
10
9
1
```

Program output

```
Input a guess:
Correct!
```

```
Input a guess:
Correct!
Input a guess:
Correct!
Input a guess:
Correct!
Input a guess:
Correct!
You have guessed all the secrets!
```

Example 2

User input & program output

```
Input a guess:
1
Correct!
Input a guess:
1
Incorrect!
Input a guess:
3
Correct!
Input a guess:
2
Incorrect!
Input a guess:
1
Incorrect!
Input a guess:
3
Incorrect!
Input a guess:
9
Correct!
Input a guess:
10
Correct!
Input a guess:
7
Correct!
You have guessed all the secrets!
```

User input

```
1
1
3
2
1
3
9
10
7
```

Program output

```
Input a guess:
Correct!
Input a guess:
Incorrect!
Input a guess:
Correct!
Input a guess:
Incorrect!
Input a guess:
Incorrect!
Input a guess:
Incorrect!
Input a guess:
Correct!
Input a guess:
Correct!
Input a guess:
Correct!
You have guessed all the secrets!
```

Task 6: All secrets with hints (30%)

This task is similar to Task 5, but the user now gets hints to help them find the secrets.

Task Description

Write a program that asks the user to guess all secret numbers of a predetermined set, stored as a python list `[3, 1, 7, 9, 10]`, and provides hints to guide the user to the closest yet-to-be-found secret.

> In this task, you may need to use a `while` loop inside another `while` loop.
>
> In the case two unfound secrets are equally close to a guess, the hint should be based on the secret that comes first in the list. For example, if the guess is `2`, and the secrets are `[3, 1]`, then the hint must be that the closest secret is greater than the guess.

Input and Output Examples

> You may temporarily change the list of secrets to find if that helps your debugging, but make sure to revert it before running automated tests, as they rely on the list being unchanged.

Example 1

User input & program output

```
Input a guess:
3
Correct!
Input a guess:
7
Correct!
Input a guess:
10
Correct!
Input a guess:
9
Correct!
Input a guess:
1
Correct!
You have guessed all the secrets!
```

User input

```
3
7
10
9
1
```

Program output

```
Input a guess:
Correct!
Input a guess:
Correct!
Input a guess:
Correct!
Input a guess:
Correct!
Input a guess:
Correct!
You have guessed all the secrets!
```

Example 2

User input & program output

```
Input a guess:
1
Correct!
Input a guess:
1
Incorrect!
The closest unknown secret is greater than your guess.
Input a guess:
3
Correct!
Input a guess:
2
Incorrect!
The closest unknown secret is greater than your guess.
Input a guess:
1
Incorrect!
The closest unknown secret is greater than your guess.
Input a guess:
3
Incorrect!
The closest unknown secret is greater than your guess.
Input a guess:
9
Correct!
Input a guess:
10
Correct!
Input a guess:
7
Correct!
You have guessed all the secrets!
```

User input

```
1
1
3
2
1
3
9
10
7
```

Program output

```
Input a guess:
Correct!
Input a guess:
Incorrect!
The closest unknown secret is greater than your guess.
Input a guess:
Correct!
Input a guess:
Incorrect!
The closest unknown secret is greater than your guess.
Input a guess:
Incorrect!
The closest unknown secret is greater than your guess.
Input a guess:
Incorrect!
The closest unknown secret is greater than your guess.
Input a guess:
Correct!
Input a guess:
Correct!
Input a guess:
Correct!
You have guessed all the secrets!
```

Example 3

User input & program output

```
Input a guess:
8
Incorrect!
The closest unknown secret is smaller than your guess.
Input a guess:
7
Correct!
Input a guess:
8
Incorrect!
The closest unknown secret is greater than your guess.
Input a guess:
3
Correct!
Input a guess:
1
```

```
Correct!
Input a guess:
10
Correct!
Input a guess:
7
Incorrect!
The closest unknown secret is greater than your guess.
Input a guess:
9
Correct!
You have guessed all the secrets!
```

User input

```
8
7
8
3
1
10
7
9
```

Program output

```
Input a guess:
Incorrect!
The closest unknown secret is smaller than your guess.
Input a guess:
Correct!
Input a guess:
Incorrect!
The closest unknown secret is greater than your guess.
Input a guess:
Correct!
Input a guess:
Correct!
Input a guess:
Correct!
Input a guess:
Incorrect!
The closest unknown secret is greater than your guess.
Input a guess:
Correct!
You have guessed all the secrets!
```

Task 8: One config & type (30%)

This task uses the same board as Task 7, but this time Robbie the robot is moving on it (imagine a big keyboard or a small Robbie)!

Task Description

Write a program that asks the user to input a string, and then plans the actions of Robbie the robot so that it can type this string on a keyboard.

The keyboard has the layout below.

```
chunk
fjord
gymps
vibex
waltz
```

The robot starts at the top left position (`c`).

The robot can take the following actions:

```
u - go up
d - go down
l - go left
r - go right
p - press the key
```

After pressing a key, Robbie stays on that key. This means:

- It can immediately press the same key again.
- To go to another key, Robbie will start from the last key it moved to.

Furthermore, Robbie will always move vertically (up/down) before moving horizontally (left/right) when moving to a key it needs to press.

Robbie's actions are restricted by the limits of the board:

- It cannot take an action that would make it leave the board.
- It cannot wrap around the board.

Input and Output Examples

Example 1

User input & program output

```
Enter a string to type:
q
The string cannot be typed out.
```

User input

```
q
```

Program output

```
Enter a string to type:
The string cannot be typed out.
```

Example 2

User input & program output

```
Enter a string to type:
chord
The robot must perform the following operations:
prpdrprprp
```

User input

```
chord
```

Program output

```
Enter a string to type:
The robot must perform the following operations:
prpdrprprp
```

Example 3

User input & program output

```
Enter a string to type:
helpiamsentient
The robot must perform the following operations:
rpdddrrpdlpuurpdllpdpuurprrpdlpuuupddddpullprrpuuupddddp
```

User input

```
helpiamsentient
```

Program output

```
Enter a string to type:
The robot must perform the following operations:
rpdddrrpdlpuurpdllpdpuurprrpdlpuuupddddpullprrpuuupddddp
```

Task 9: Many configs (50%)

This task uses the same movement rules as Task 8, but this time Robbie has multiple keyboard configurations to choose from, and will choose the one that requires the fewest moves.

Task Description

Write a program that asks the user to input a string, then choose a keyboard, then plans the actions of Robbie the robot so that it can type this string on a keyboard.

For an input string,

- Robbie must pick one keyboard and only use this keyboard,
- Robbie will always choose the keyboard configuration that requires the fewest moves out of all keyboard configurations that allow that string to be typed.
- You can assume that there is never a tie between two or more keyboards,
- There may not be a keyboard that can type that string.

The keyboards have the configurations below.

Configuration 0

```
abcdefghijklm
nopqrstuvwxyz
```

Configuration 1

```
789
456
123
0.-
```

Configuration 2

```
bemix
clunk
grypt
vozhd
waqfs
```

Configuration 3

```
chunk
fjord
gymps
```

```
vibex
waltz
```

Robbie starts at the top left position of the chosen keyboard.

Input and Output Examples

Example 1

User input & program output

```
Enter a string to type:
q
Configuration index:
0
The robot must perform the following operations:
drrrp
```

User input

```
q
```

Program output

```
Enter a string to type:
Configuration index:
0
The robot must perform the following operations:
drrrp
```

Example 2

User input & program output

```
Enter a string to type:
chord
Configuration index:
3
The robot must perform the following operations:
prpdrprprp
```

User input

```
chord
```

Program output

```
Enter a string to type:
Configuration index:
3
The robot must perform the following operations:
prpdrprprp
```

Example 3

User input & program output

```
Enter a string to type:
areyoustillthere
Configuration index:
2
The robot must perform the following operations:
ddddrpuupuupddrpdlpuurpdddrrpuupuulpdllppdrrrpdlpuuullpddpuup
```

User input

```
areyoustillthere
```

Program output

```
Enter a string to type:
Configuration index:
2
The robot must perform the following operations:
Ddddrpuupuupddrpdlpuurpdddrrpuupuulpdllppdrrrpdlpuuullpddpuup
```

Task 10: Many configs & wrap (20%)

This task is similar to Task 9, but this time Robbie can "wrap" around the keyboard, which means cross the border to the other side of the keyboard.

Task Description

Write a program that asks the user to input a string, then choose a keyboard, then plans the actions of Robbie the robot so that it can type this string on a keyboard.

This time, though, Robbie is able to wrap around the keyboard. For example:

- In the row "abcde", Robbie can move from 'a' to 'e' in 1 move to the left. This move is encoded as "lw", where "w" marks a wrap (horizontal or vertical).
- In the row "abcde", Robbie can move from 'e' to 'a' in 1 move to the right. This move is encoded as "rw".
- The same applies for columns. The character 'w' is also used to mark vertical wrapping.
- Proud of this new technique, and for style points, Robbie uses wrapping whenever it does not increase the number of moves required. This means that, for example, in the row "ab", to go from 'a' to 'b', Robbie will always prefer doing "lw" than "r" ('w' does not count as a move).

The other rules and keyboard configurations are the same as in Task 9.

> For a given input string, the best keyboard may not be the same as for Task 9, now that wrapping is possible.

Input and Output Examples

Example 1

User input & program output

```
Enter a string to type:
q
Configuration index:
2
The robot must perform the following operations:
uwrrp
```

User input

```
q
```

Program output

```
Enter a string to type:
Configuration index:
2
The robot must perform the following operations:
uwrrp
```

Example 2

User input & program output

```
Enter a string to type:
nm
Configuration index:
0
The robot must perform the following operations:
uwpdwlwp
```

User input

```
nm
```

Program output

```
Enter a string to type:
Configuration index:
0
The robot must perform the following operations:
uwpdwlwp
```

Example 3

User input & program output

```
Enter a string to type:
areyoustillthere
Configuration index:
3
The robot must perform the following operations:
uwrpdwdrrpddpullpurpupddrrpddlpullpdrpprpdwllpuwurrpuupddp
```

User input

```
areyoustillthere
```

Program output

```
Enter a string to type:
Configuration index:
3
The robot must perform the following operations:
uwrpdwdrrpddpullpurpupddrrpddlpullpdrpprpdwllpuwurrpuupddp
```

Task 11: Rabbytes' Age (40%)

Fibonacci is having a hard time keeping track of his fast-multiplying rabbytes. He has requested your help to create a database of rabbytes.

Task Description

Write a program that allows a user to create and store new rabbytes. Each rabbit is identified by a unique name. The program also allows the user to enter a rabbit's age.

Main menu

The program prompt of the main menu looks like:

```
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
```

If the user inputs 1, 2, 3 or 0, it runs the corresponding function. Otherwise, the prompt is simply printed again. See example 1.

> We recommend you create a function for each menu option.

1. Create a Rabbit

Upon inputting `1`, the program will then print the prompt

```
Input the new rabbit's name:
```

If the user enters a new name, the program returns to the menu and prints the main prompt.

Otherwise, the program prints

```
That name is already in the database.
Input the new rabbit's name:
```

and repeat this until the user inputs a new name. See example 2.

2. Input Age of a Rabbit

Upon inputting `2`, the program will then print the prompt

```
Input the rabbit's name:
```

If the user inputs a name (e.g. `rabbie`) that is already in the database, the program prompts

```
Input the rabbit's name:
rabbie
Input rabbie's age:
2
```

It then returns to the main menu and prints the main prompt.

If the user inputs a name that does not exist, the program prints

```
That name is not in the database.
Input the rabbit's name:
```

until an existing name is provided. See example 3.

> We assume that:
> - the age input is an integer,
> - the user will only pick option 2 if a rabbit has already been created,
> - the user can freely set the age of a rabbit, even if it already has one.

3. List Rabbytes

Upon inputting `3`, the program will then print

```
Rabbytes:
rabbie (42)
not_rabbie (Unknown)
```

if two rabbytes, `rabbie` and `not_rabbie`, have been created before, and if `rabbie`'s age (42) has been input, and `not_rabbie`'s age hasnt. See example 4.

> Display the rabbytes in the order they were created.

Examples

User inputs are in **bold font** below.

Example 1

```
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
```

```
3. List Rabbytes.
0. Quit.
==================================
foo
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
0
```

Example 2

```
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
1
Input the new rabbit's name:
undercover_robbie
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
1
Input the new rabbit's name:
undercover_robbie
That name is already in the database.
Input the new rabbit's name:
robbie_the_rabbit
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
0
```

Example 3

```
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
1
Input the new rabbit's name:
rabbie
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
2
Input the rabbit's name:
bob
That name is not in the database.
Input the rabbit's name:
rabbie
Input rabbie's age:
2
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
0
```

Example 4

```
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
3
Rabbytes:
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
1
Input the new rabbit's name:
rabbie
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
1
Input the new rabbit's name:
not_rabbie
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
2
Input the rabbit's name:
rabbie
Input rabbie's age:
42
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
3
Rabbytes:
rabbie (42)
not_rabbie (Unknown)
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
0. Quit.
==================================
0
```

Task 12: Rabbytes' Lineage (30%)

Impressed with your work, Fibonacci wants you to improve your program to track the lineage of his wacky rabbytes.

Task Description

Write a program that, on top of the previous options, allows the user to specify a parent/kitten relationship.

Main menu

The main prompt now has an extra two options:

```
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
0. Quit.
==================================
```

> We recommend you create a function for each menu option, and reuse functions you have written (by improving them if need be) if possible.

4. Create a Parental Relationship

This menu allows the user to indicate a parental relationship between two rabbytes, by first inputting the name of the parent, then the name of the kitten.

```
Input the parent's name:
Anakin
Input the kitten's name:
Luke
```

If either name does not correspond to an existing rabbit, a rabbit with that name is created, as it would be if that name was provided through option 1 of the menu. See example 1.

> You do not need to verify any consistency of the input. In particular, you may suppose that the user and the automated tests will:
> - ensure that the number of parents is between 0 and 2.
> - a rabbit cannot be their own ancester.
> However, make no other assumptions as to how rabbytes reproduce.

5. List Direct Family of a Rabbit

This menu asks the user to enter a name, and then displays its parents and kittens.

```
Input the rabbit's name:
Anakin
Parents of Anakin:
Kittens of Anakin:
Leia
Luke
```

If the input name is not a known rabbit, the program prints

```
That name is not in the database.
Input the rabbit's name:
```

until a known name is provided. The parents and kittens are then displayed as above. See example 2.

> Assume that the user (and the automated tests) will only pick option 5 if a rabbit has already been created.
>
> Display the names of the parents and the kittens in alphabetical order. You may use Python's built-in sorting utilites.

Examples

User inputs are in **bold font** below.

Example 1

```
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
0. Quit.
==================================
4
Input the parent's name:
Anakin
Input the kitten's name:
Luke
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
0. Quit.
```

```
==================================
1
Input the new rabbit's name:
Leia
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
0. Quit.
==================================
4
Input the parent's name:
Anakin
Input the kitten's name:
Leia
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
0. Quit.
==================================
3
Rabbytes:
Anakin (Unknown)
Luke (Unknown)
Leia (Unknown)
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
0. Quit.
==================================
5
Input the rabbit's name:
Anakin
Parents of Anakin:
Kittens of Anakin:
Leia
Luke
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
0. Quit.
==================================
5
Input the rabbit's name:
Leia
Parents of Leia:
```

```
Anakin
Kittens of Leia:
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
0. Quit.
==================================
0
```

Example 2

```
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
0. Quit.
==================================
4
Input the parent's name:
Anakin
Input the kitten's name:
Leia
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
0. Quit.
==================================
5
Input the rabbit's name:
Luke
That name is not in the database.
Input the rabbit's name:
Leia
Parents of Leia:
Anakin
Kittens of Leia:
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
0. Quit.
==================================
0
```

Task 13: Rabbytes' Extended Universe (30%)

Just when you thought your program met Fibonacci's expectations, he barges into your room, out of breath, and screams "they are out of control"! He explains that they are nearing their critical mass (a terabbyte), and that the impending catastrophe can only be avoided if your program helps him isolate rabbit cousins.

Task Description

Main menu

The main prompt now has an extra option:

```
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
6. Find Cousins of a Rabbit.
0. Quit.
==================================
```

> We recommend you create a function for each menu option, and reuse functions you have written (by improving them if need be) if possible.

6. Find Cousins of a Rabbit

This menu allows the user to list the cousins of a rabbit. A cousin is a kitten of a sibling of a parent.

```
Input the rabbit's name:
cousin1
Cousins of cousin1:
cousin2
```

Similar to menu option 5, it asks the user for a name again

```
That name is not in the database.
Input the rabbit's name:
```

until it corresponds to one in the database. See example 1.

> Display the names of the cousins in alphabetical order. You may use Python's built-in sorting utilites.

A rabbit cannot be its own cousin. Every other rabbit can.

Examples

User inputs are in **bold font** below.

Example 1

```
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
6. Find Cousins of a Rabbit.
0. Quit.
==================================
4
Input the parent's name:
grandma
Input the kitten's name:
dad1
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
6. Find Cousins of a Rabbit.
0. Quit.
==================================
4
Input the parent's name:
grandma
Input the kitten's name:
dad2
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
6. Find Cousins of a Rabbit.
0. Quit.
==================================
4
Input the parent's name:
dad1
Input the kitten's name:
cousin1
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
```

```
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
6. Find Cousins of a Rabbit.
0. Quit.
==================================
4
Input the parent's name:
dad2
Input the kitten's name:
cousin2
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
6. Find Cousins of a Rabbit.
0. Quit.
==================================
6
Input the rabbit's name:
cousin1
Cousins of cousin1:
cousin2
==================================
Enter your choice:
1. Create a Rabbit.
2. Input Age of a Rabbit.
3. List Rabbytes.
4. Create a Parental Relationship.
5. List Direct Family of a Rabbit.
6. Find Cousins of a Rabbit.
0. Quit.
==================================
0
```