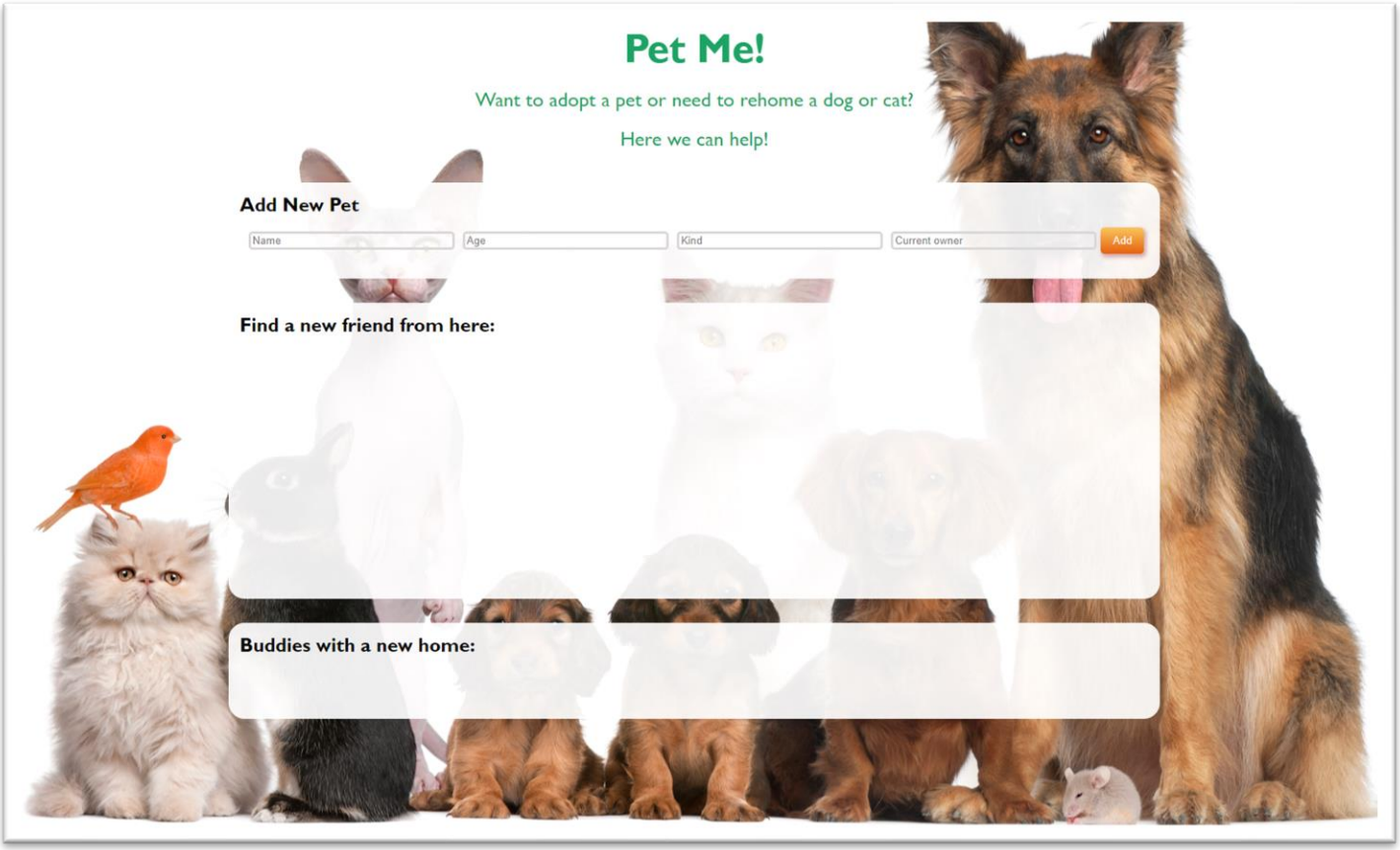
JS Advanced: Regular Exam 27.06.2020

Problem 1. Pet Me (DOM Manipulation)

Use the **given skeleton** to solve this problem.

Note: You have **NO permission** to change directly the given HTML *(index.html file)*.



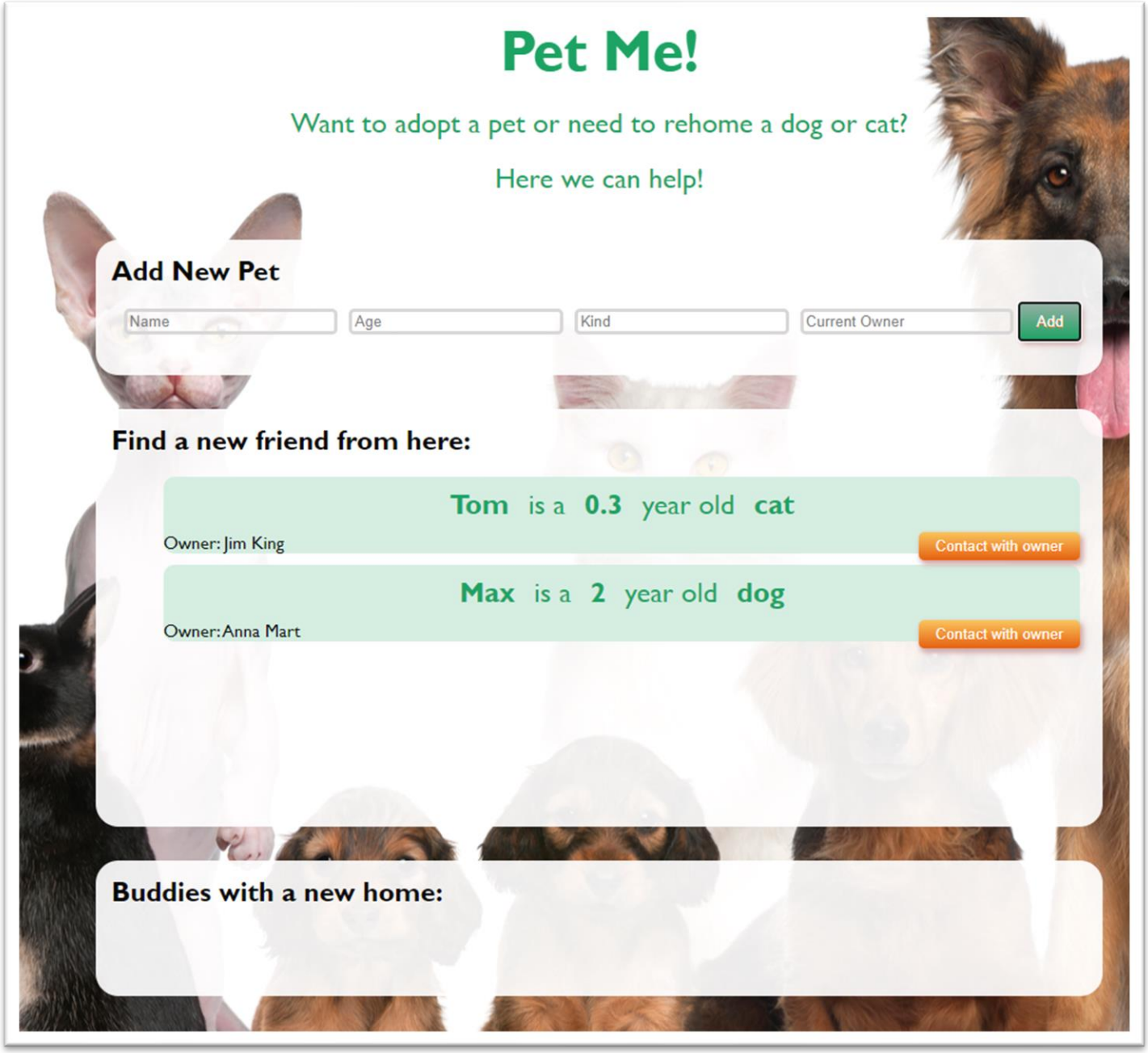
Your Task

Write the missing JavaScript code to make the **Pet Me** application work as expected.

Each new registered pet must have **`Name`**, **`Age`**, **`Kind`** and **`Current Owner`**.

When you click the **`[Add]`** button and **only** if all **inputs** are **filled** and the age is a **number**, then a new **list item** should be **added** to the section with id "**`adoption`**". Don't forget to **clear the inputs** when you add a new pet.

Already added pets



The new item should have the **following structure**:

```
▼<section id="adoption">
    <h2>Find a new friend from here:</h2>
  ▼<ul>
    ▼<li>
      ▼<p> == $0
          <strong>Tom</strong>
          " is a "
          <strong>0.3</strong>
          " year old "
          <strong>cat</strong>
        </p>
        <span>Owner: Jim King</span>
        <button>Contact with owner</button>
      </li>
    ▶<li>…</li>
    </ul>
  </section>
```

You should create a **`li`** element that contains **`paragraph`** element with the name, age and kind of the new pet, follow the format "**`{ name } is a { years } year old { kind }`**", where **name, years** and **kind** are in a **`strong`** elements inside the paragraph. After that we have **span** element with "**`Owner: { owner name }`**" and a button **`[Contact with owner].`**

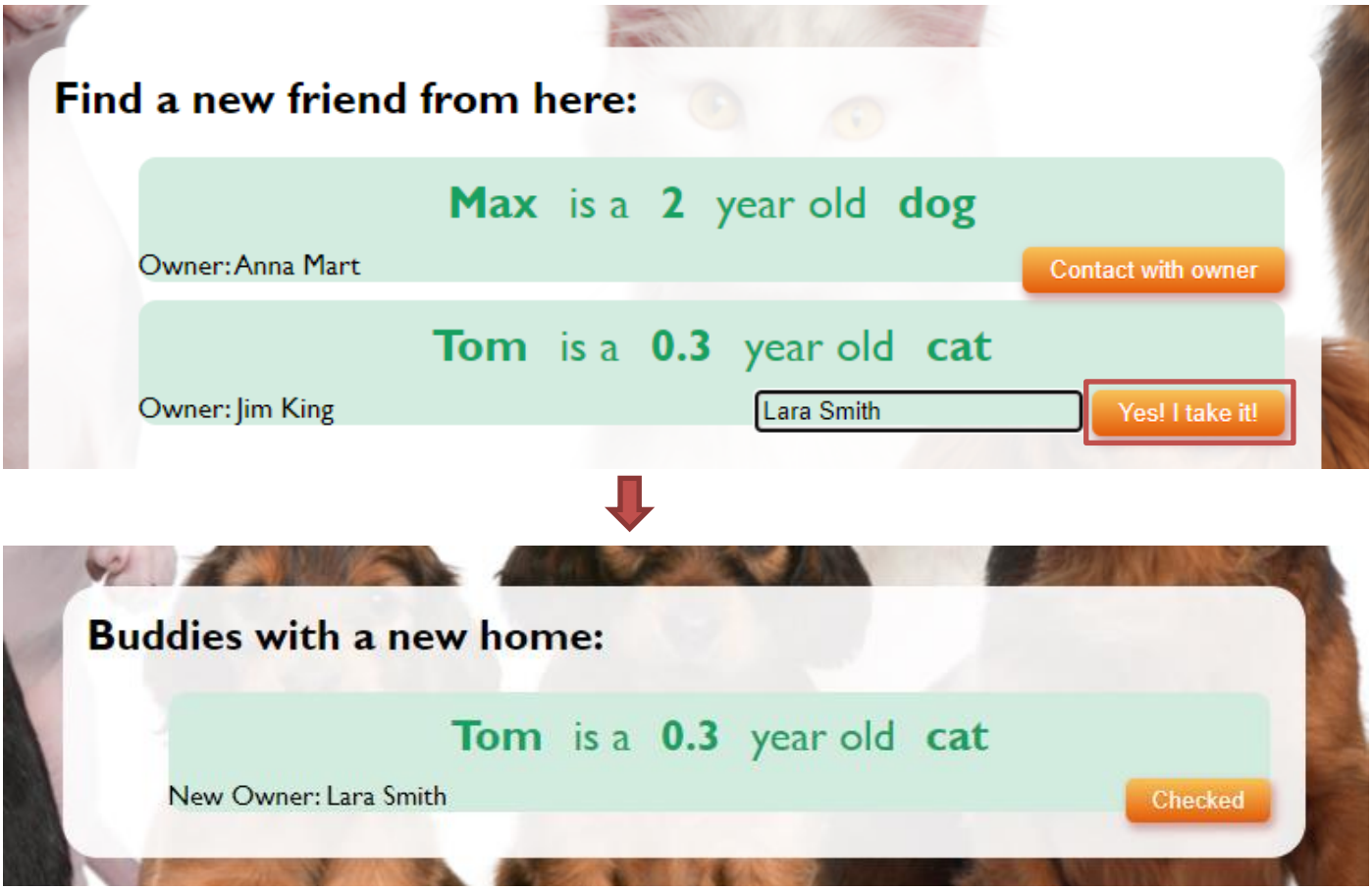
When you click the **`[Contact with owner]`** button an input appears and the button changes to **`[ Yes! I take it! ]`** like this:

The new elements are into a **`div`** element and structure is changed like this:

```
▼<ul>
  ▼<li>
    ▼<p> == $0
      <strong>Tom</strong>
      " is a "
      <strong>0.3</strong>
      " year old "
      <strong>cat</strong>
    </p>
    <span>Owner: Jim King</span>
    ▼<div>
      <input placeholder="Enter your names">
      <button>Yes! I take it!</button>
    </div>
  </li>
  ▶<li>…</li>
</ul>
```

Moving pets into the new home section

When you click the **`[Yes! I take it!]`** button **if there is entered name** you should **move the current list item** to the **adopted section.**



Here we have changed the **owner name** with the new one. And the button is **[Checked]**. We have the next HTML structure:

```
▼<section id="adopted">
    <h2>Buddies with a new home:</h2>
  ▼<ul>
    ▼<li>
      ▼<p> == $0
          <strong>Tom</strong>
          " is a "
          <strong>0.3</strong>
          " year old "
          <strong>cat</strong>
        </p>
        <span>New Owner: Lara Smith</span>
        <button>Checked</button>
      </li>
    </ul>
  </section>
```

And in the end button **[Checked]** must **delete** the current list item.

Problem 2. Pet House

Your Task

Implement the following classes: **Pet, Cat, Dog.**

Class Pet

constructor(owner, name)

Should have these **3** properties:

- **owner – string;**
- **name – string;**
- **comments - array;**

addComment(comment)

This **function** should receive single **comment** like **string**, add it to the **comments** array and **return** a message:

"Comment is added."

If comment is **already added** to the comments array **throw** an **error** with:

"This comment is already added!"

feed()

This **function** should **return** a simple message:

"{ name } is fed"

toString()

This **function** should **return string:**

```
"Here is { owner }'s pet { name }."
```

If there are any **comments** then add on a new line:

```
"Special requirements: { comment1 }, { comment2 }, { comment3 ...}"
```

Class Cat

Class **Cat** inherits class **Pet**.

constructor(owner, name, insideHabits, scratching)

Should have these **4** properties:

- **owner – string,**
- **name – string,**
- **insideHabits – string,**
- **scratching - boolean,**

feed()

This **function** should inherit the **feed()** method of class **Pet** and extend the **returned** message adding this at the same line at the end:

```
", happy and purring."
```

toString()

This **function** should extend the **toString()** method of class **Pet**, **returning** the **message** with some more lines at the end which are**:**

```
"Main information:
{ name } is a cat with { insideHabits }"
```

And if **scrathing** property is **true** you should add this at the end:

```
", but beware of scratches."
```

Note: **For more information see the examples below!**

Class Dog

Class **Dog** inherits class **Pet**.

constructor(owner, name, runningNeeds, trainability)

Should have these **4** properties:

- **owner – string,**
- **name – string,**
- **runningNeeds – string**
- **trainability - string**

feed()

This **function** should inherit the **`feed()`** method of class **`Pet`** and extend the **returned message** adding this at the end:

`", happy and wagging tail."`

toString()

This **function** should extend the **`toString()`** method of class **`Pet`** returning the message with some more lines at the end which are:

`"Main information:`

`{ name } is a dog with need of { runningNeeds }km running every day and { trainability } trainability."`

Note: **For more information see the examples below!**

Submission

Submit your **`solveClasses`** function.

```
JS petHouse.js > solveClasses

function solveClasses() {

    // TODO...

    return {
        Pet,
        Cat,
        Dog
    }
}
```

Examples

This is an example how the code is **intended to be used**:

Sample code usage

`let classes = solveClasses();`
`let pet = new classes.Pet('Ann', 'Merry');`
`console.log(pet.addComment('likes bananas'));`
`console.log(pet.addComment('likes sweets'));`
`console.log(pet.feed());`
`console.log(pet.toString());`
 	
`let cat = new classes.Cat('Jim', 'Sherry', 'very good habits', true);`
`console.log(cat.addComment('likes to be brushed'));`
`console.log(cat.addComment('sleeps a lot'));`

```
console.log(cat.feed());
console.log(cat.toString());

let dog = new classes.Dog('Susan', 'Max', 5, 'good');
console.log(dog.addComment('likes to be brushed'));
console.log(dog.addComment('sleeps a lot'));
console.log(dog.feed());
console.log(dog.toString());
```

Corresponding output

```
Comment is added.
Comment is added.
Merry is fed
Here is Ann's pet Merry.
Special requirements: likes bananas, likes sweets

Comment is added.
Comment is added.
Sherry is fed, happy and purring.
Here is Jim's pet Sherry.
Special requirements: likes to be brushed, sleeps a lot
Main information:
Sherry is a cat with very good habits, but beware of scratches.

Comment is added.
Comment is added.
Max is fed, happy and wagging tail.
Here is Susan's pet Max.
Special requirements: likes to be brushed, sleeps a lot
Main information:
Max is a dog with need of 5km running every day and good trainability.
```

Problem 3. Veterinary Clinic

```
class VeterinaryClinic {
    // TODO: implement this class...
}
```

Your Task

Write a **`class VeterinaryClinic`**, which implements the following functionality:

Functionality

constructor (clinicName, capacity)

Receives **2** parameters at initialization of the class (**clinicName** and **capacity**).

Should have these **3** properties:

- **`clinicName`** – property of type string;
- **`capacity`** – property of type number;
- **`clients`** – initially an empty array;

Hint: Add more properties like **`totalProfit`** and **`currentWorkload`** to help you finish the task. Read the problem description **until the end** to get clear with the requirements.

newCustomer(ownerName, petName, kind, procedures)

The **`ownerName`**, **`petName`** and **`kind`** are of type **string** and the **`procedures`** are an **array** of strings. This information will be used in the following **`toString()`** method.

`ownerName` – **string** that keeps the name of the current pet owner, one owner **may have more than one pets** under his name, choose customer structure wisely to collect all of the given information. Once stored this information stays in the clinic data, even when the pet is healed.

`petName` – **string** that keeps the name of the current pet, once stored this information stays in the clinic data, even when the pet is healed.

`kind` – **string** that keeps the current pet kind, be **careful** of **upper cases** into the input string. Once stored this information stays in the clinic data, even when the pet is healed.

`procedures` – **array** of strings that keeps the procedures the current pet kind needs. **You know that a pet is a current client when one or more procedures are recorded at his list of procedures.** When pet is healed and leaves the clinic the array of procedures must be emptied. So when the pet comes back again for healing it can be listed with new procedures.

Before register:

- Check if the clinic is able to accept more pets. If the clinic is full **throw an `Error`**:

```
"Sorry, we are not able to accept more patients!"
```

- Check if the pet is already registered under this client name. If it's registered and still has full list of procedures, you should **throw an `Error`**:

```
"This pet is already registered under { ownerName } name! { petName } is on our lists,
waiting for { all his procedures separated by ', ' }."
```

- Otherwise this function should **add** the customer and his pet, **update** the current clinic **workload** and **return:**

```
"Welcome { petName }!"
```

onLeaving (ownerName, petName)

- Check if the given **ownerName** corresponds to a client in the **clients** array, if not **throw an Error**:

  "Sorry, there is no such client!"

- Then check if the given **petName** is **registered** under this **ownerName**, if not or it is **registered but the procedures array is empty because all his procederues are done** , then **throw an Error**:

  "Sorry, there are no procedures for { petName }!"

- Otherwise, on leaving you have to **collect** the current client bill, add it to the total **clinic profit** and **save the data**. Calculate the bill knowing that each **procedure** cost **500.00$.** Do not forget to **update** the current **workload**. Clear the array with **procedures** for the current pet.

  When pet leaves the clinic, **petName** and **kind** should stay like information in our data, but with **no more** procedures in the **array of procedures**. After that **return,** the following string:

  "Goodbye { petName }. Stay safe!"

toString ()

Return the full information of the clinic.

- First, we have to **calculate** how busy the clinic is in **percentages**. Percentage represents all current **pets** that have **procedures** based on the **full capacity** of the clinic. The percentage is rounded to the nearest smaller number:

  "{ clinicName } is { percentage }% busy today!"

- On the **second** line comes the collected **profit,** that must be **fixed** to the **second digit** after the decimal point**:**

  "Total profit: { profit }$"

  On the next lines, return the whole information for the owners in the following format**.** Print **kind** property with **lowercase** letters. All owners should be in **alphabetical order**, as also pets of each of them must be in **alphabetical order** too:

  "{ ownerName } with:

  ---{ petName } - a { kind } that needs: { procedures separated by ', '}"

Examples

This is an example how the code is **intended to be used:**

Sample code usage

```
let clinic = new VeterinaryClinic('SoftCare', 10);
console.log(clinic.newCustomer('Jim Jones', 'Tom', 'Cat', ['A154B', '2C32B', '12CDB']));
console.log(clinic.newCustomer('Anna Morgan', 'Max', 'Dog', ['SK456', 'DFG45', 'KS456']));
console.log(clinic.newCustomer('Jim Jones', 'Tiny', 'Cat', ['A154B']));
console.log(clinic.onLeaving('Jim Jones', 'Tiny'));
console.log(clinic.toString());
clinic.newCustomer('Jim Jones', 'Sara', 'Dog', ['A154B']);
console.log(clinic.toString());
```

Corresponding output

```
Welcome Tom!
Welcome Max!
Welcome Tiny!
Goodbye Tiny. Stay safe!
SoftCare is 20% busy today!
Total profit: 500.00$
Anna Morgan with:
---Max - a dog that needs: SK456, DFG45, KS456
Jim Jones with:
---Tiny - a cat that needs:
---Tom - a cat that needs: A154B, 2C32B, 12CDB
SoftCare is 30% busy today!
Total profit: 500.00$
Anna Morgan with:
---Max - a dog that needs: SK456, DFG45, KS456
Jim Jones with:
---Sara - a dog that needs: A154B
---Tiny - a cat that needs:
---Tom - a cat that needs: A154B, 2C32B, 12CDB
```

GOOD LUCK! 😊