



UPU, Freshman campus

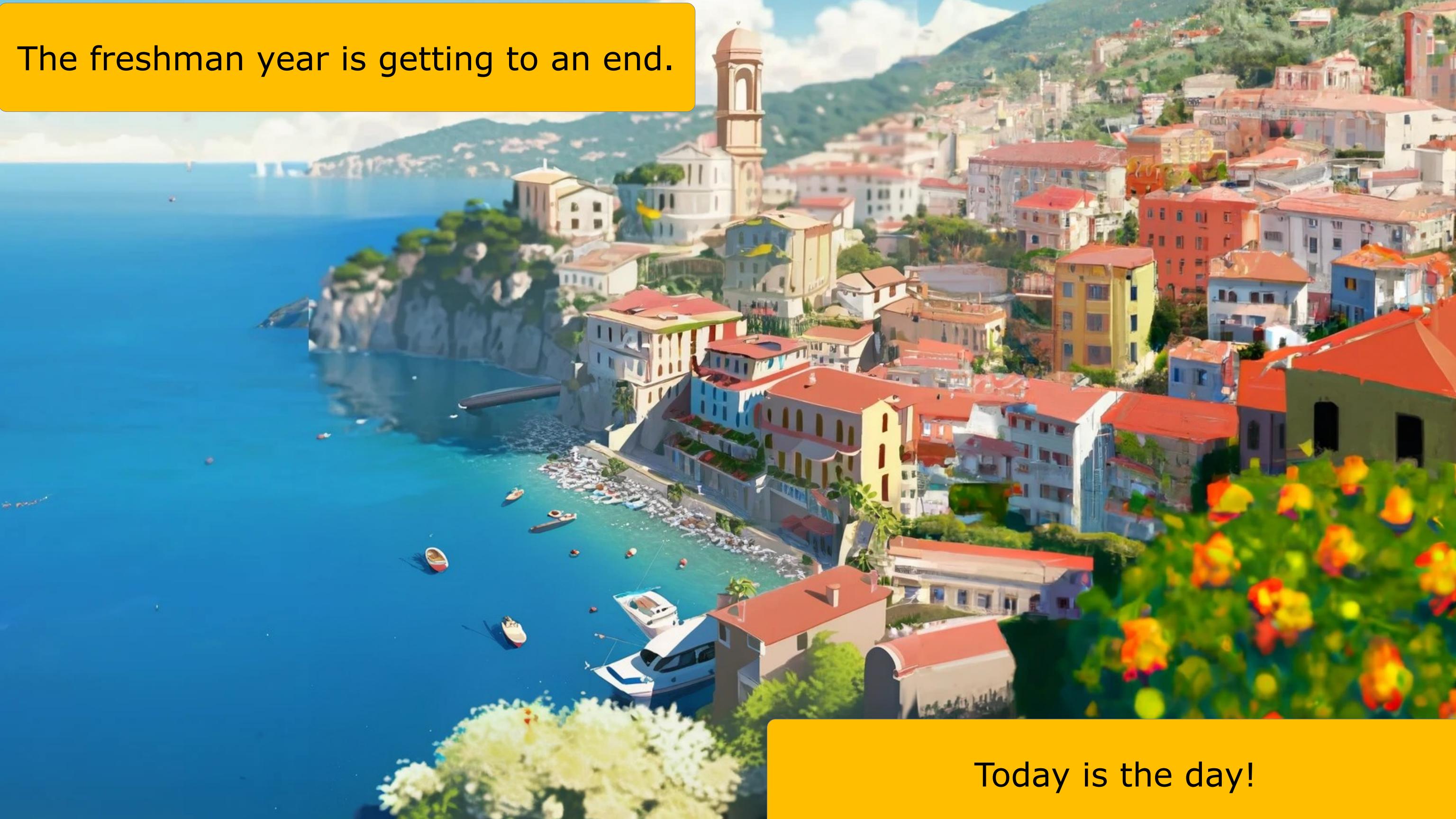




The freshman year is getting to an end.



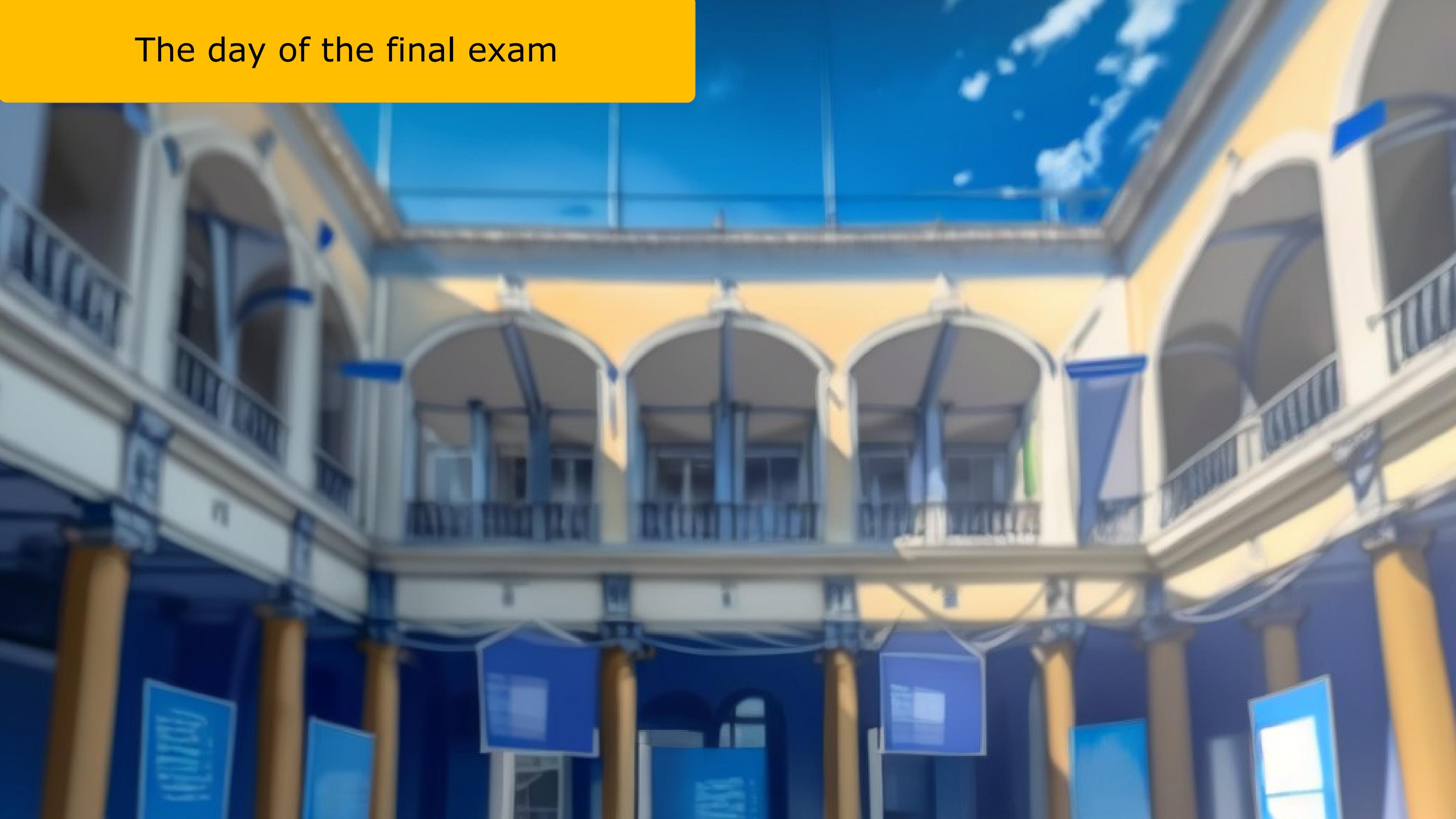
The freshman year is getting to an end.

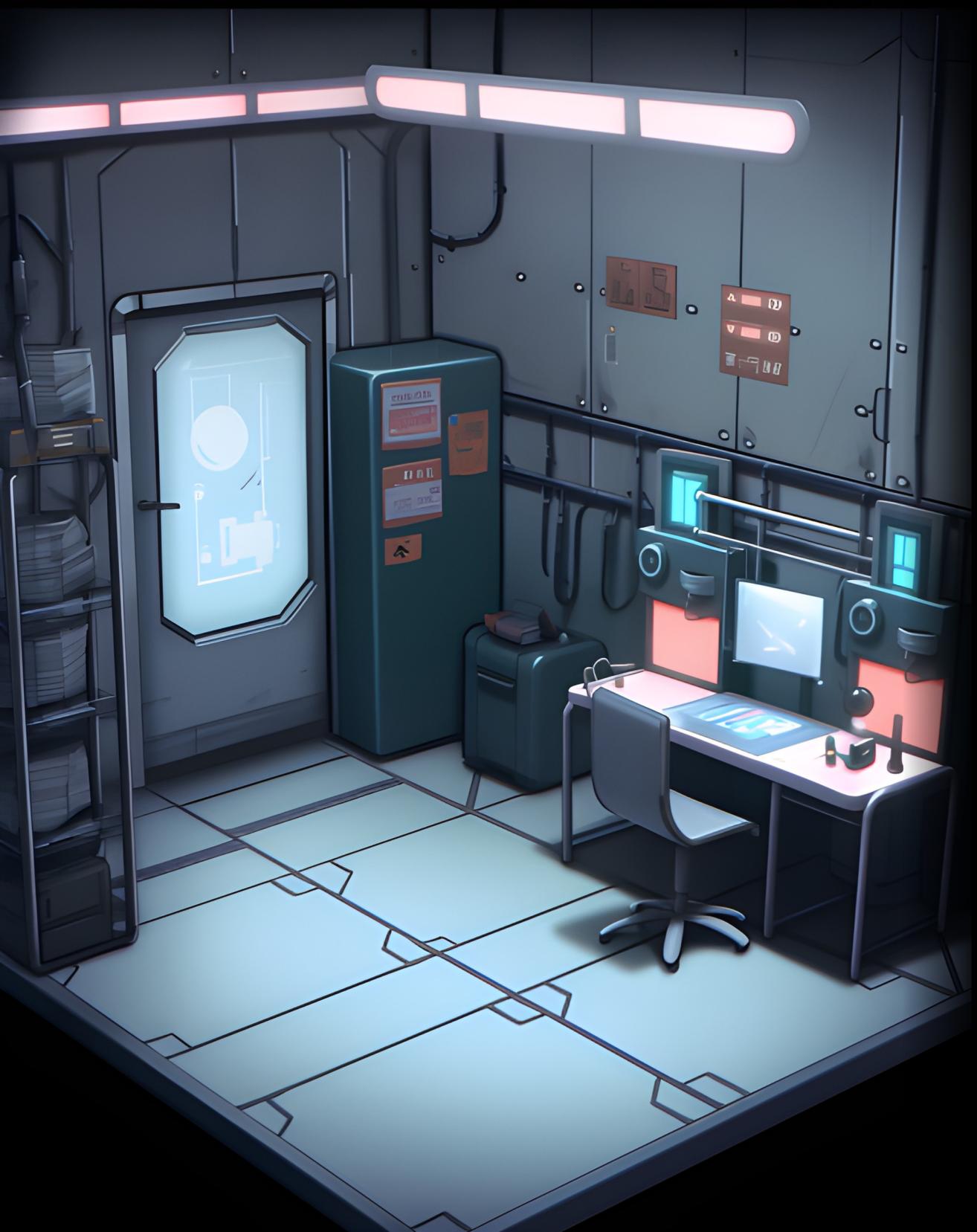


Today is the day!

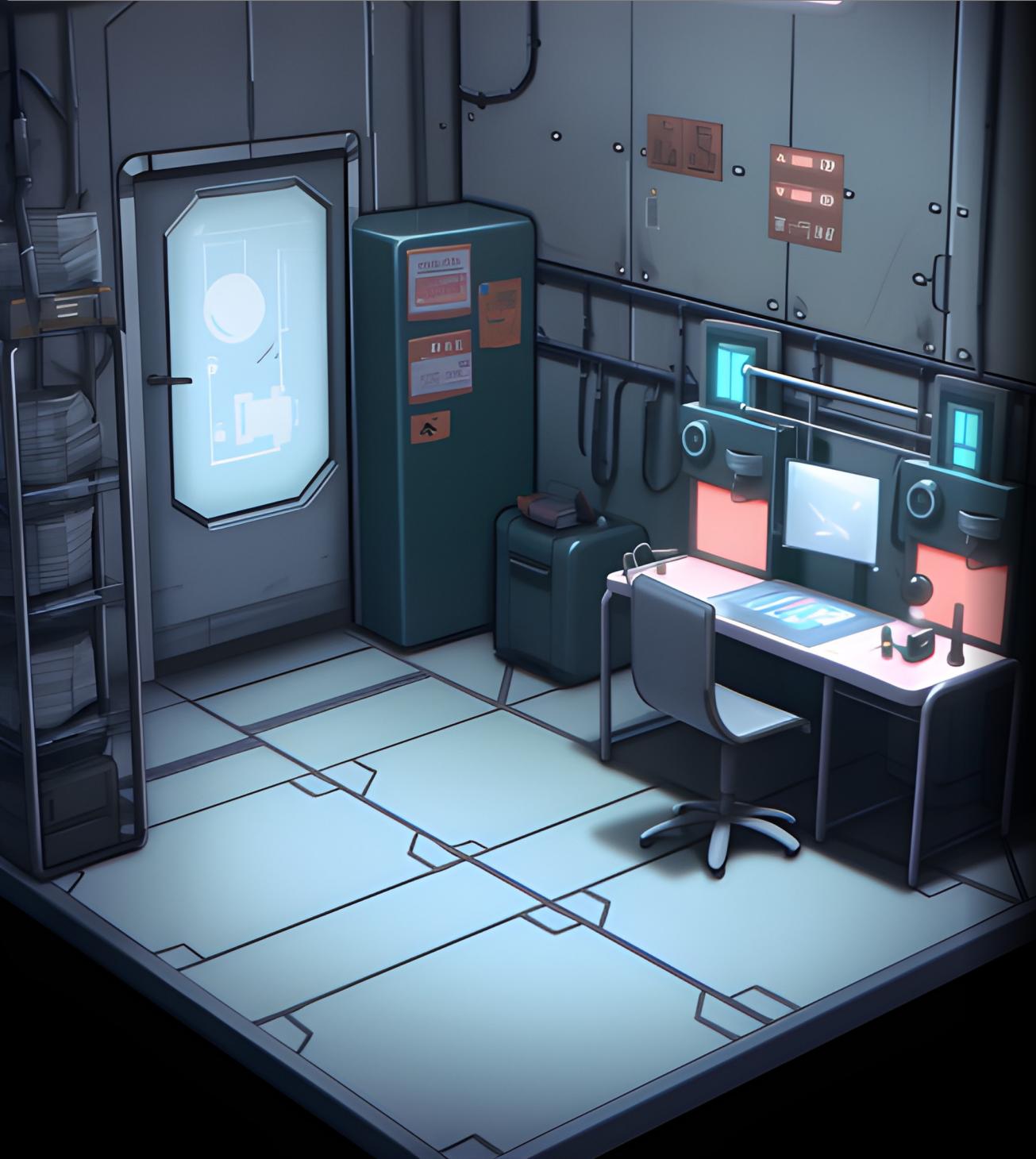


The day of the final exam





The final exam starts at 9am.



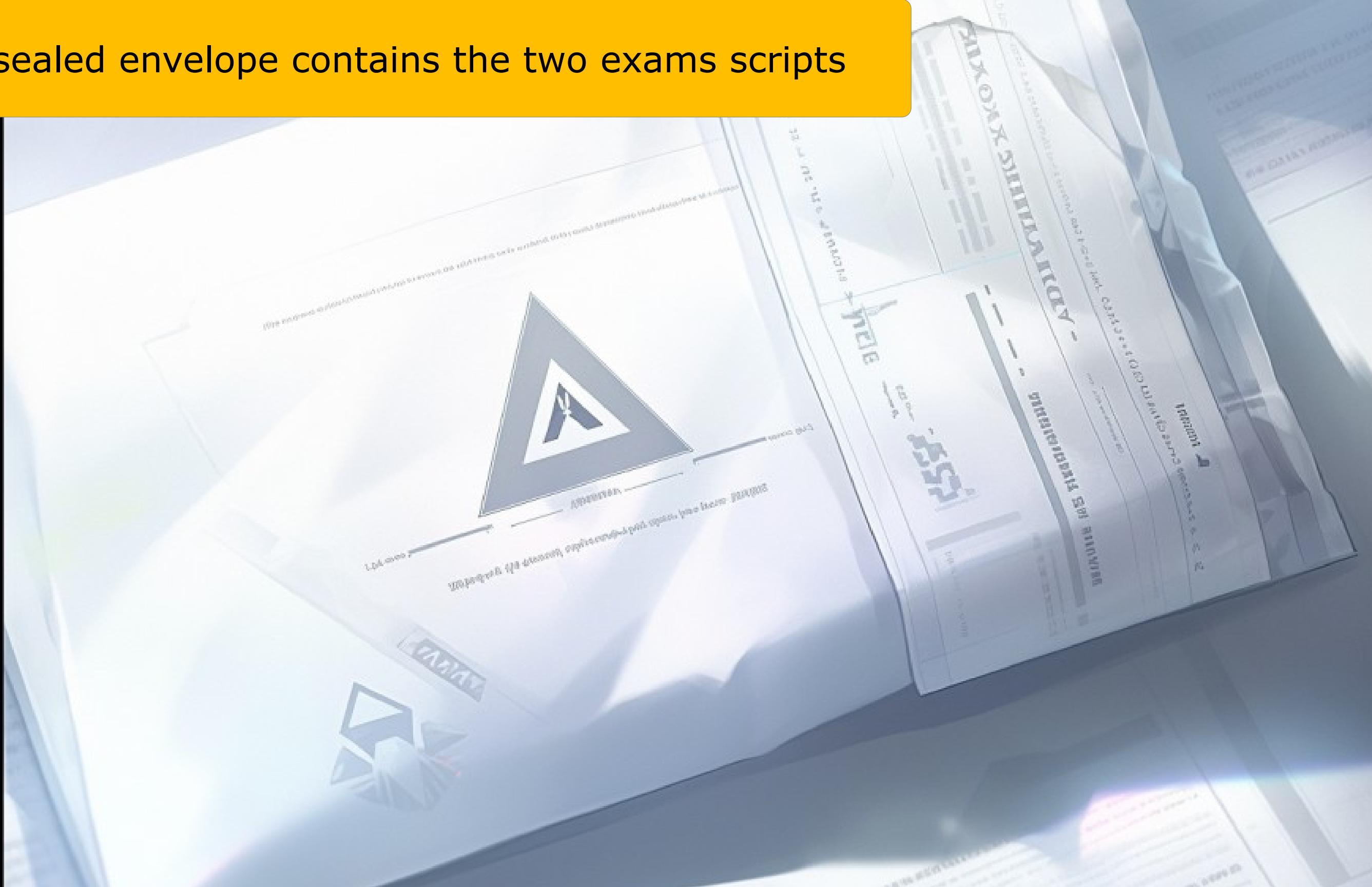
The final exam starts at 9am.

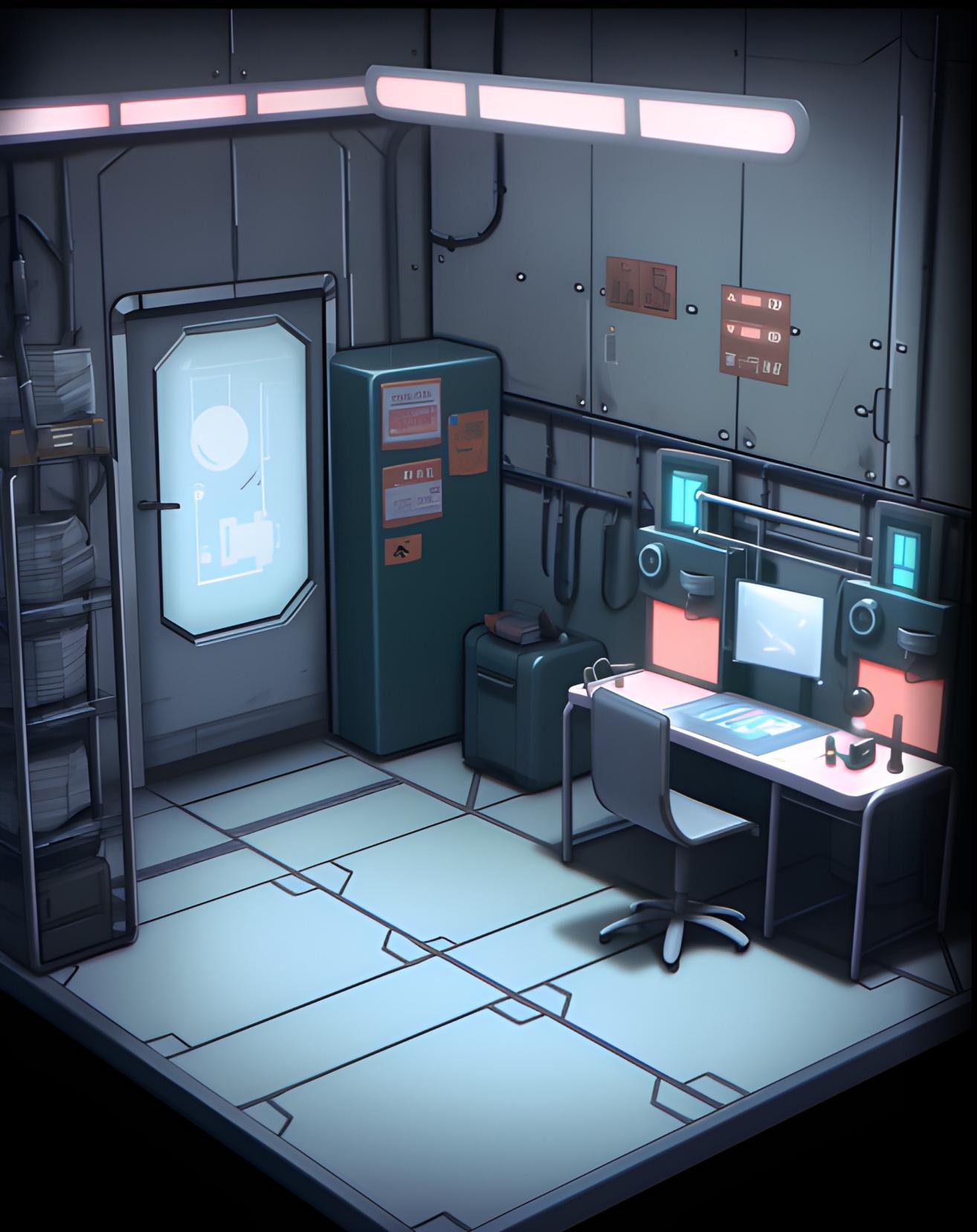


The exam is held in sealed individual room and it lasts 5 hours.



A sealed envelope contains the two exams scripts



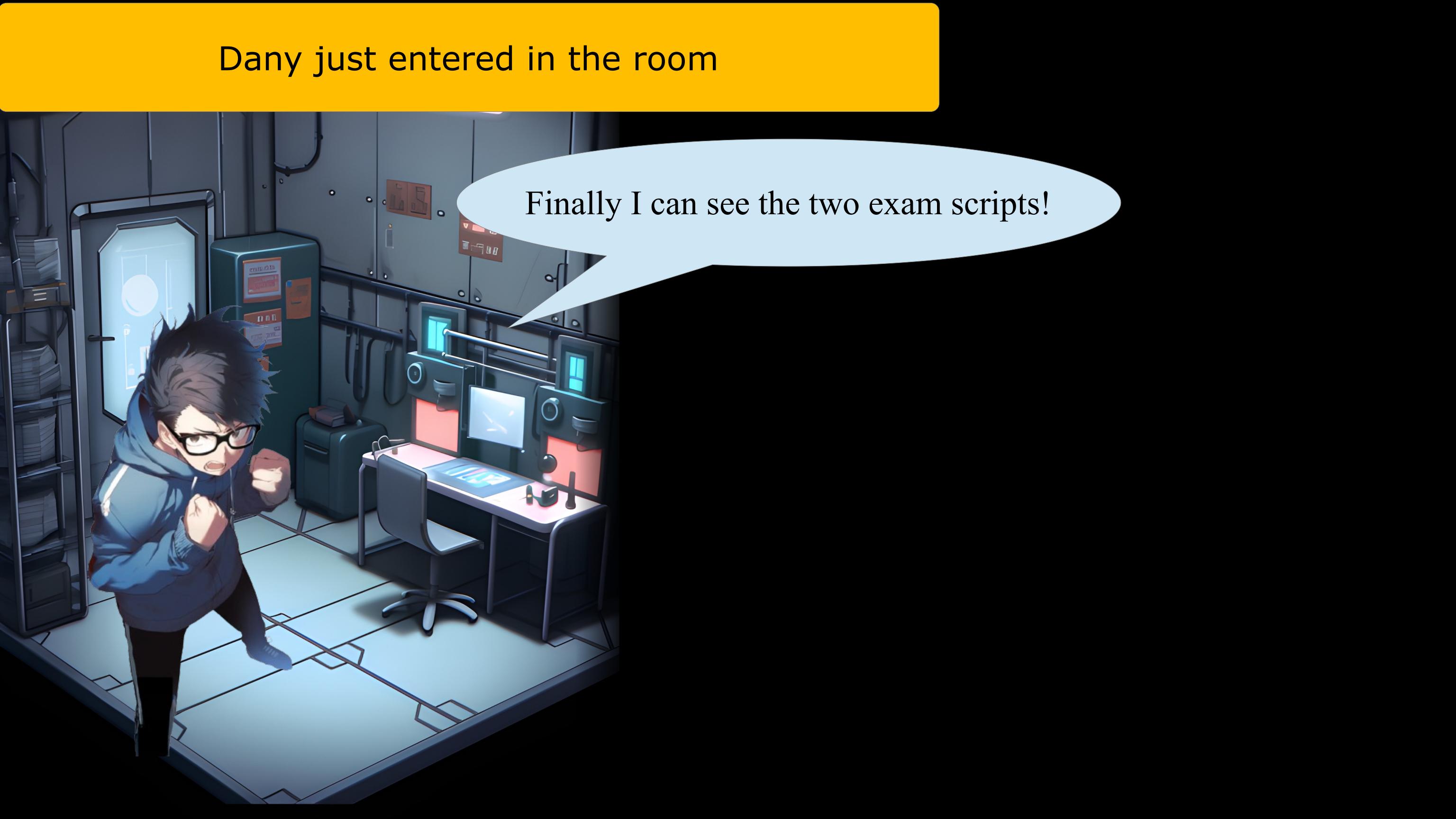




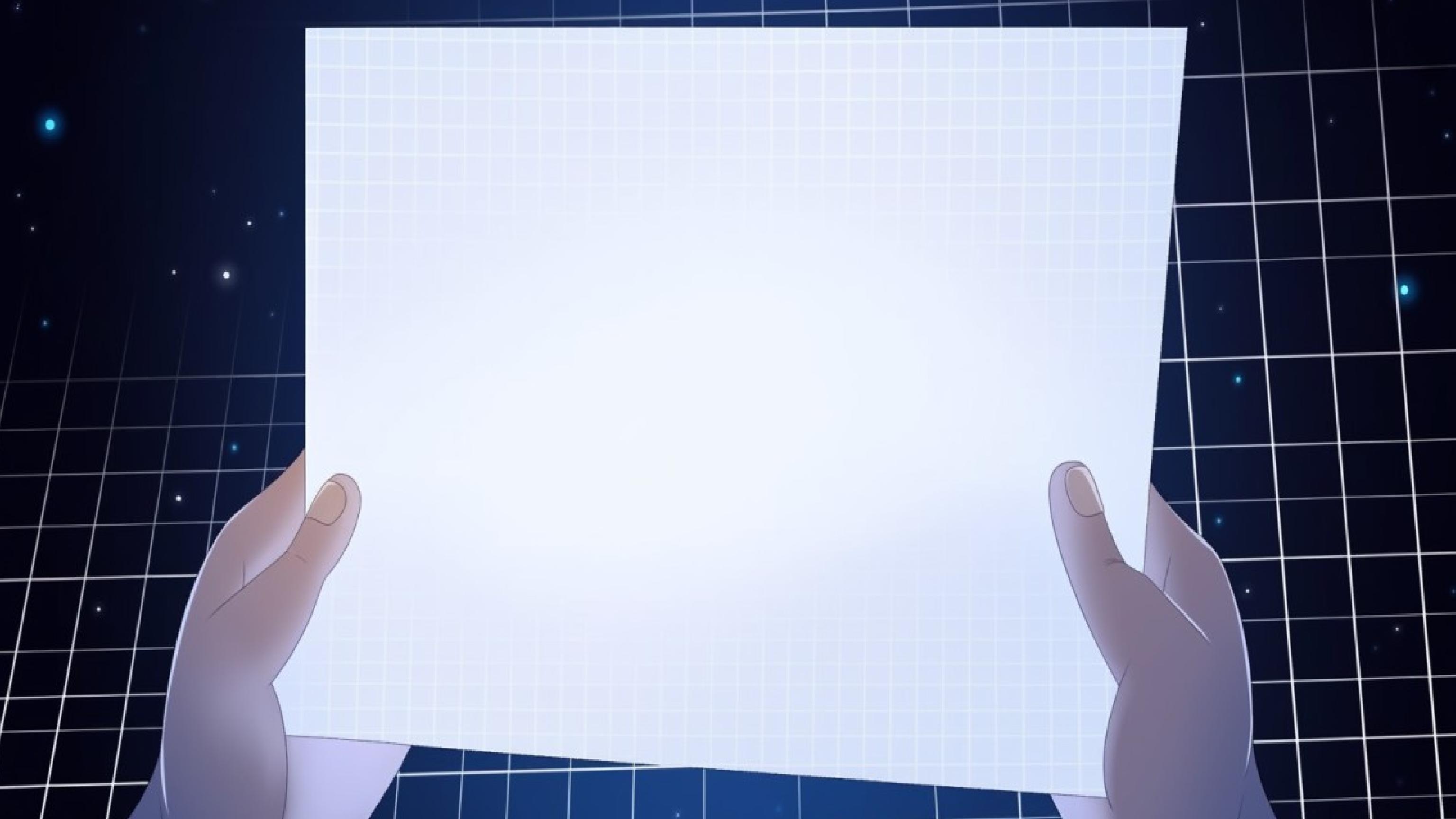
Dany just entered in the room



Dany just entered in the room

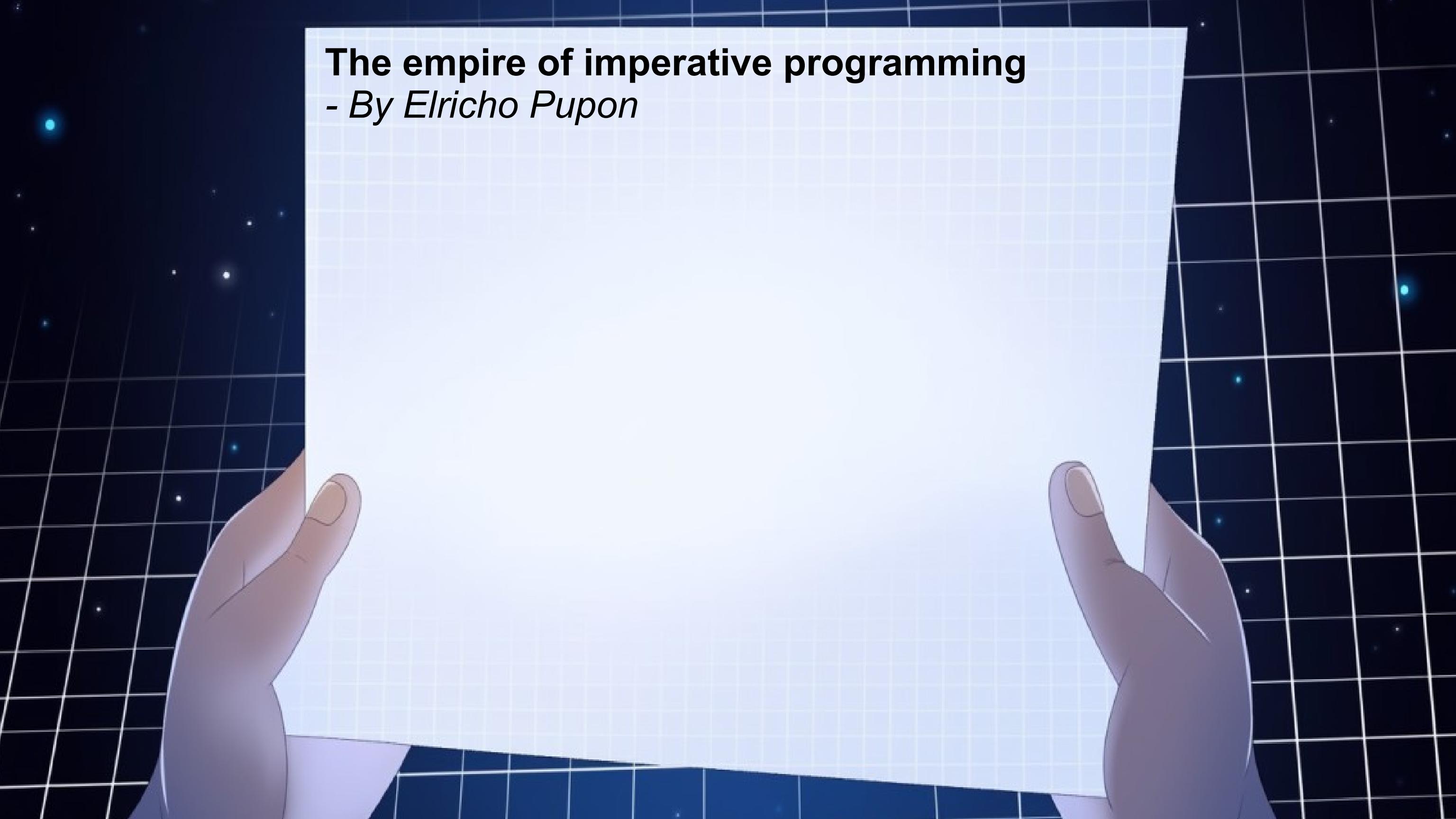


Finally I can see the two exam scripts!



The empire of imperative programming

- By *Elricho Pupon*



The empire of imperative programming

- By *Elricho Pupon*

Implement an efficient and elegant merge sort algorithm in Java.

Details: the algorithm must take in input a list of an appropriate type T, and must produce a sorted version of that list, without modifying the original.

The empire of imperative programming

- By Elricho Pupon

Implement an efficient and elegant merge sort algorithm in Java.

Details: the algorithm must take in input a list of an appropriate type T, and must produce a sorted version of that list, without modifying the original.

The algorithm should run in $O(n * \log n)$, and allocate no more than $O(n)$ space. It should rely on high level abstractions instead of indexes.





Merge sort! Yes, I remember it!
It is about dividing a list in half,
over and over again.



Merge sort! Yes, I remember it!
It is about dividing a list in half,
over and over again.

Eventually, the list parts become singleton lists:
lists with just one element.



Merge sort! Yes, I remember it!
It is about dividing a list in half,
over and over again.

Eventually, the list parts become singleton lists:
lists with just one element.

And a list with a single element
is intrinsically sorted!



Merge sort! Yes, I remember it!
It is about dividing a list in half,
over and over again.

Eventually, the list parts become singleton lists:
lists with just one element.

And a list with a single element
is intrinsically sorted!

It is about dividing your problems into smaller
and smaller problems until they disappear!





A large, detailed red dragon with a textured, scaly body and a long, spiny tail is shown from the waist up, facing right. It has a wide, toothy mouth and a small horn on its forehead. To the right of the dragon is a white speech bubble containing the text. A man with short brown hair and black-rimmed glasses is partially visible on the far right, wearing a dark blue suit jacket.

A giant list is
a giant problem to solve

A large, detailed red dragon with a textured, scaly body and a single visible blue eye. It has a wide, toothy mouth and a small horn on its forehead. A speech bubble originates from its mouth.

A giant list is
a giant problem to solve

A close-up of a man's face wearing black-rimmed glasses. He has a thoughtful expression with his hand near his chin. A speech bubble originates from his mouth.

An overwhelming
problem!







If we can somehow
divide the problem in two





If we can somehow
divide the problem in two

We would have two
smaller disasters!





If we can somehow
divide the problem in two

We would have two
smaller disasters!



We can then attack
them one at the time, and
recover between them



A large, red, muscular dragon-like creature with a speech bubble.

If the two new problems are
still too big to attack directly

A large, red, scaly dragon-like creature with a textured, bumpy skin surface. It has a wide, toothy mouth, two small horns on its head, and a long, spiny tail. A white speech bubble is positioned above its head, containing text.

If the two new problems are
still too big to attack directly

A person seen from behind, wearing a blue hoodie. A white speech bubble is positioned above their head, containing text.

We can just recursively
divide them over
and over again







They still look menacing





They still look menacing

We can divide
them more!







Now they look
so numerous





Now they look
so numerous



But in programming,
this is not a problem.



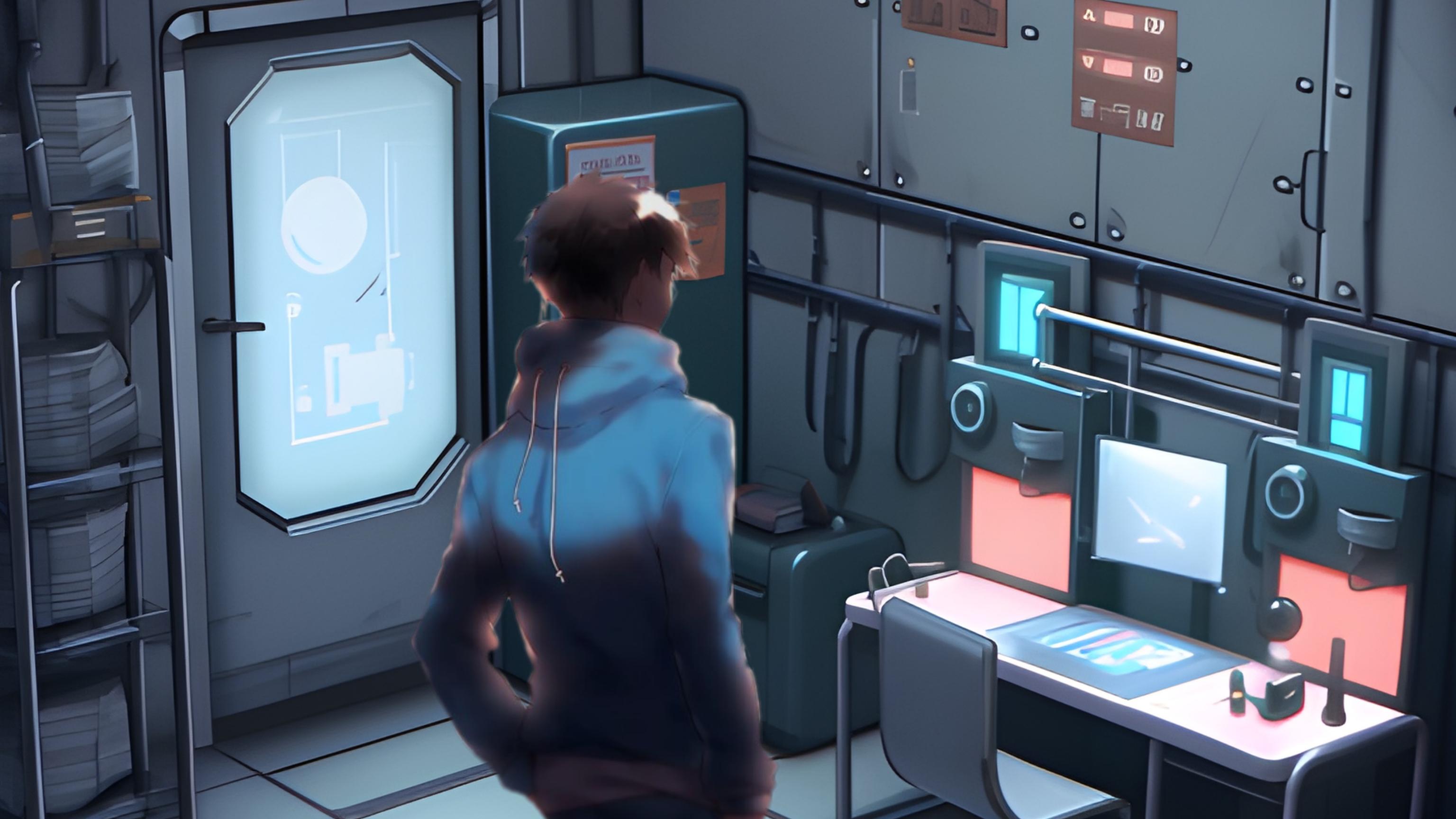
Now they look
so numerous



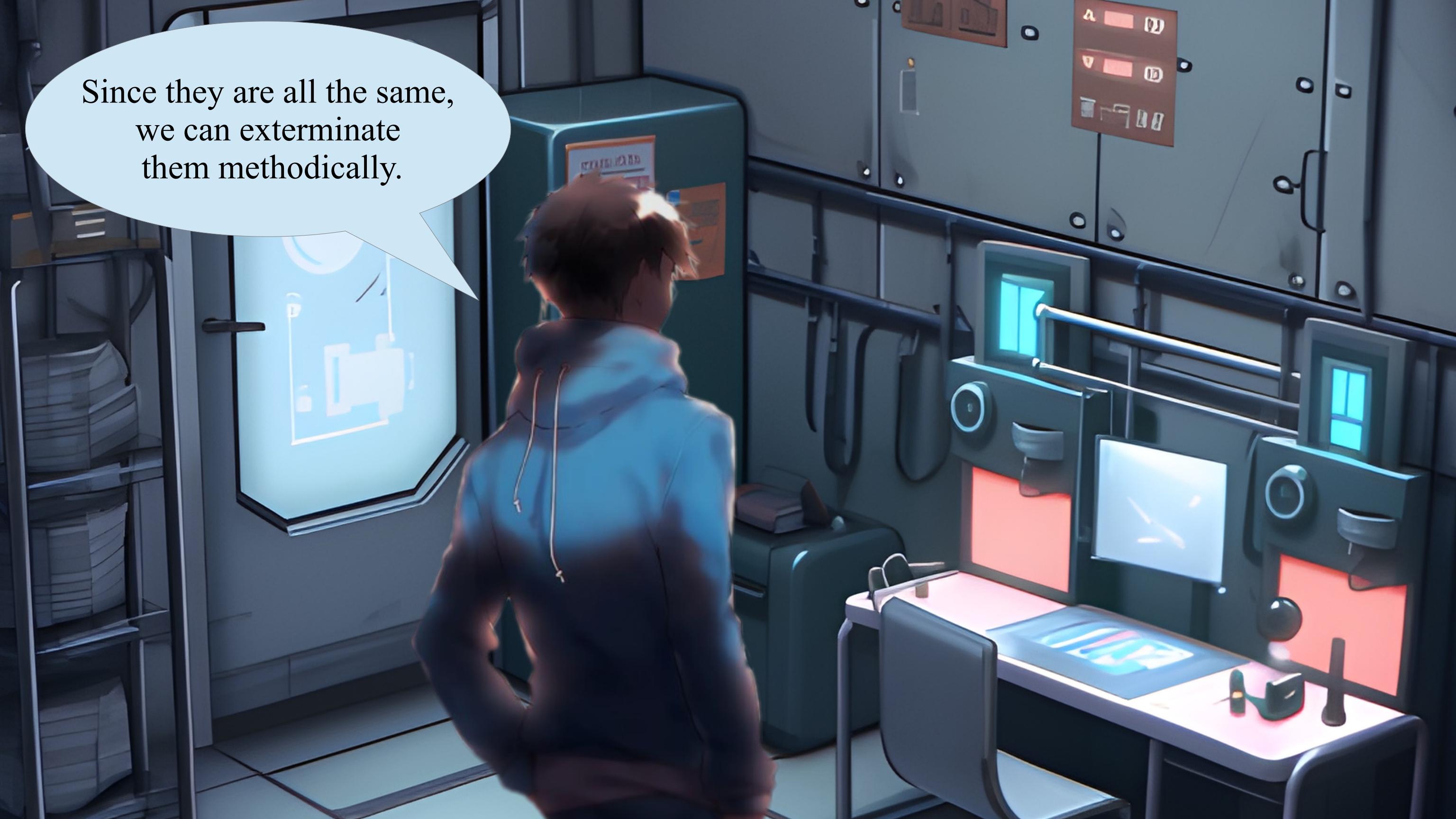
But in programming,
this is not a problem.



They are many, but they
are all the same!



Since they are all the same,
we can exterminate
them methodically.





Since they are all the same,
we can exterminate
them methodically.

We can call the same
method over and over again
to squash them all in
the same way







In the case of sorting,
squashing them
is not even needed.





In the case of sorting,
squashing them
is not even needed.



A list with a single element
is intrinsically sorted.













After solving all
the individual
sub problems





After solving all
the individual
sub problems

We need to do one last thing.



After solving all
the individual
sub problems

We need to do one last thing.



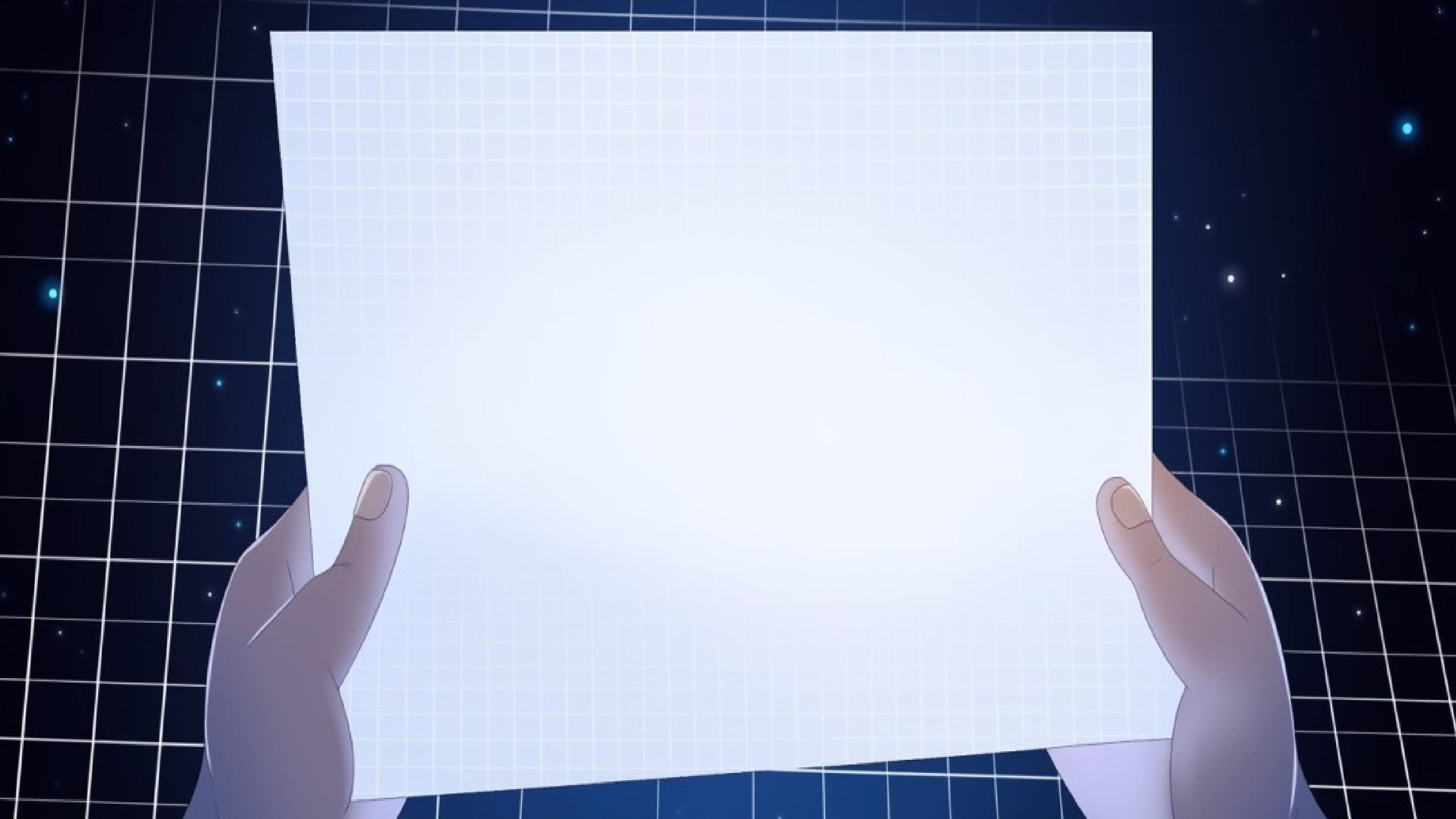
Merge solved solutions
into larger solution.



I think I can do it!

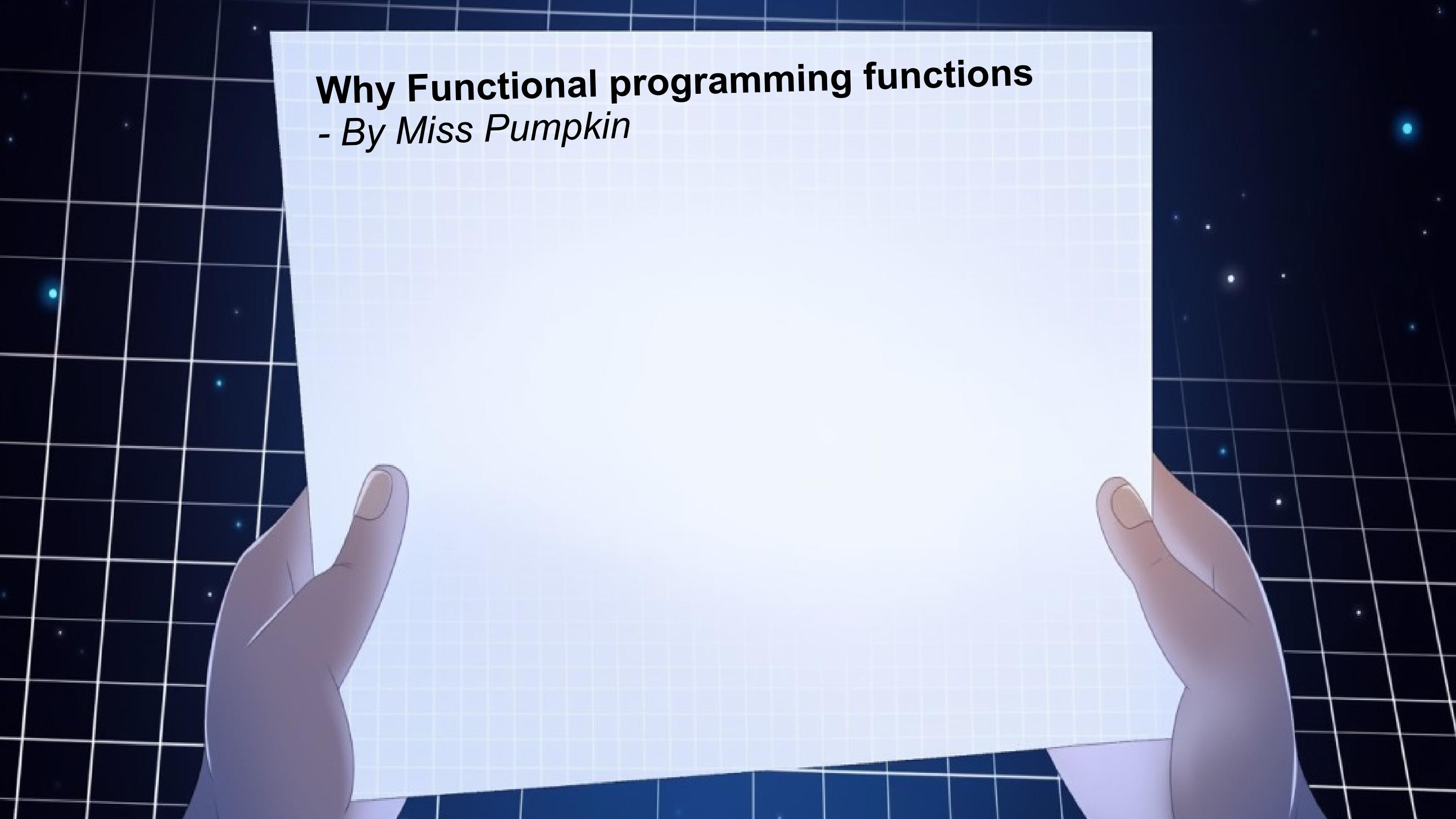
Let see the second script





Why Functional programming functions

- By *Miss Pumpkin*



Why Functional programming functions

- By Miss Pumpkin

Re-implement the Optional type in Java from scratch.

You must use only pure OO features.

Carefully chose what operations are the most important to show.

Make sure to handle Serialization properly.





Yes, I think I can do this too!



Yes, I think I can do this too!

***Hi Dany, Mewice
to see you again.***



*I'm CAT, the
computerized
automated tutor*





*I'm CAT, the
computerized
automated tutor*

*I'm here to
supervise you*



*I'm CAT, the
computerized
automated tutor*

*I'm here to
supervise you*

*... and to guide you
in submitting your
answers*





CAT is here?



CAT is here?

I've got a bad feeling about this!