Line 5:

When coding in typescript, camel Casing is the standard naming convection

when naming functions and variables, although you have used camel casing

through the rest of your code.

Your generic <T> should be placed after the equals sign.

Line 13 and 15:

In your functions parameters you have declared shift’s type as a string where

it should be an number type, as you are conducting mathematical operations on

that variable.

Line 19:

In order to use the .length() function, on the 'string' variable you will need to

convert its type on line 5 to a String type.

Line 33:

Due to the shift variable’s type being string, you will not be able to carry out the

maths operations on this line, i suggest changing its type on line 5 to number.

Line 51:

You are using 'print()', where you should be using "console.log()"

**\*\* The positive aspects of the submissions**

The logic is correct to create a Caesar Cypher. The task is almost complete, make the necessary corrections and your code should run properly.

**\*\*\* Documentation**

Although you have provided comments in your code which gives the reader an idea of what is happening, I would suggest adding more descriptive comments which will provide the user with a better understanding of your code and the logic behind it.

**\*\*\* Style**

Your comments give the read a fairly good idea of how your code works, it is structed in a clear manner and well organised. The curly braces placement, indentation and spacing is consistent throughout the code. There was no instances of code being repeated, there were also no long lines of code, although line 23 was the longest, it was still a readable length. The code within the while loop was broken into smaller sections which makes it more readable. The code resembled a well maintained and consistent coding style.

**\*\*\* Efficiency**

There are no unnecessary functions or variables used within the code and no variables which have been declared and not used. You have also only used if-else statements when necessary and only when required. Within your while loop you have also provided a break statement which will prevent an infinite loop. Overall the code is efficient.

**Frontend**

In order to run on your local machine, navigate to the client folder, which contains the frontend code and type in the following commands to install the node packages and run the development server.

**Backend**

In order to run on your local machine, navigate to the server folder, which contains the backend code and type in the following commands to install the node packages and run the development server.