Let's begin.

To code we need to create:

1. Token class: will return the token (integer, 3)
2. Interpreter class
3. interpreter class will take input, get next token
4. expr and its helper function

//Corner cases  
Whitespace error example: 4 4 + 5

Step 1

* **Creating the main function**

Now in the main function, we need to continuously ask for input like calc>  
Isme one thing is ki agar khuch input nhi kiya then again ask for input – continue  
But ab question is how to know ki we have reached finish?  
toh like python let's keep quit as our mechanism to stop the interpreter from running.

* **Next we'll send the input to our interpreter (which will be a class)**

why class? because it has to have a few functions it needs to call and it would be better to have a new object for each input given by the user. Matlab see agar normal functions banate na toh hume harr function defination ke liye arguments pass karne padte, but ek class banane se that work gets easy. As ek object ke jo variables hote voh directly use kar sakte class functions mei.

* **Deciding the structure of interpreter class**

Variables:  text, pos(determines konsa char dekh rahe hum), current token(of Token class)   
Constructor: pos = 0, text with input  
Error function: taaki koi issue ho toh we could just call this error function (rather than giving particular errors)  
Eval function: that will actually evaluate the expression  
Now we know eval function needs 1. next token function 2. eat function (basically a function which accepts a value, i.e ki uski type theek hai na matlab)

* **Now making the logic for eval function**

current token mei phele toh token read karenge (which will be by get next token function)  
will store its value int left int and then call eat to check its type, same will be done for operator and second int  
then do left+right and return the result

* **Now making the Token class**

Ab kyunki hume Tokens se deal karna, ki is the type correct hai ya nhi, ya fir next Token kaise le  
So now for the token class:  
 Variables: value(this will be of auto type as we could have int or char in this), type to store type in string.  
 We will make a constructor, which initializes the value of value and type  
 Now we need to define that if we print the Token, how will it show, like return Token karte ko kya aana chahiye  
 Auto nhi bna due toh some issue with c++, so we decided let's keep it int, it will store the ASCII values  
Also isme humne << ko overload kiya because we wanted if the next token needs to be printed, dhang se print ho jaaye voh

* **Making eat & get next token function**

Toh ab eat mei toh simply we will check the type, if current token type matches with given token type then it is right, and we can go on to have the next token.  
For get next token, phele we check the base condition ki kahi End of Line pe toh nhi aa gaye  
Agar nhi toh we check ki digit hai ya plus sign and uske according we create a Token object and return it to eval/eat function, otherwise error

Now, this completes the full implementation as mentioned in the blog, but now we have to deal with the exercises as well as the edge cases.