**Task 3 – An explanation of task 2 and its main functionality**

In task 2 our goal was to write a program in JavaScript that sends one or more formulas, received from an users input, to an API for simplification. For this task we installed node on our computer for running the JS program in the terminal with **node task2.js,** and had to run a couple imports:

import fetch from "node-fetch";

import readline from "readline";

We then prompt the user and save his input in a variable. For multiple formulas we will iterate over all of them and send each of them to our fetchFunc() function. This is an async function that will handle the GET requests to our API. A function being async means that we can write programs with asynchronous behaviour, and the async function will always return a promise. Using async function we can easily write code to interact with APIs using promises. Calling a async function within a for loop, means that the for loop will run through all its iterations without waiting, and a fetch for each formula will be sent, whereas the fastest one will be returned first.

We use multiple awaits in the program, and await can only be used inside a async function. Using await, we call fetch on the API for each formula. The await statement is used when we want the code to wait for a promise to be returned. When the fetch is called using await, the code will get a response from the API. We will then use promise, and the response will be sent to .then(). We then check if the response was successful using a different method called handleFetchError(). If the response was not ok, we catch an error. If not, we proceed.

The returned data will then be formatted nicely and shown to the user in the terminal. We now have a functional program that will send the users input to the API, and retrieve the correct result asynchronously. There will be no waiting for fetch operations to be finished, and the fastest operation will be returned first.