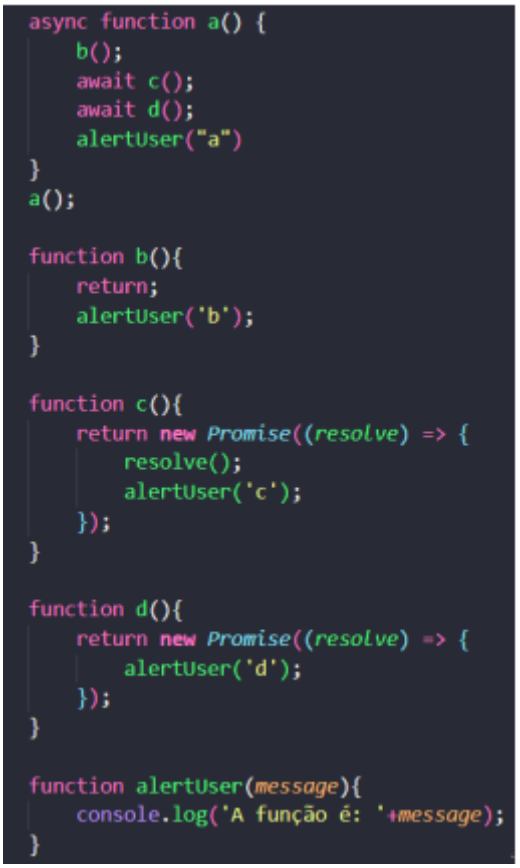

```
async function a() {
    b();
    await c();
    await d();
    alertUser("a")
}
a();

function b(){
    return;
    alertUser('b');
}

function c(){
    return new Promise((resolve) => {
        resolve();
        alertUser('c');
    });
}

function d(){
    return new Promise((resolve) => {
        alertUser('d');
    });
}

function alertUser(message){
    console.log('A função é: '+message);
}
```

Bônus:

3) Qual a ordem dos prints no console?

"A função é: c" - Isso ocorre porque a função c() é a primeira a ser chamada com await. Ela cria uma promessa que é resolvida imediatamente e, em seguida, exibe "c" no console.

"A função é: d" - A função d() é chamada logo após c(), mas não usa await. Portanto, ela exibe "d" no console imediatamente.

4) Existe algum erro nesse código? Se sim, comente sobre?

Não existem erros que travem o código ou impeçam de executá-lo, porém há algumas coisas que podem ser aprimoradas ou esclarecidas.

1 - Função b(): Esta função contém um return antes da chamada a alertUser('b'). Isso significa que a função b() retornará imediatamente, e a chamada a alertUser('b') nunca será executada.

2 - Funções c() e d(): Essas funções retornam promessas, mas elas não têm nenhum código assíncrono real dentro delas. Nesse contexto, as promessas são resolvidas imediatamente, o que não parece ter um propósito prático.

3 - Uso de await: A função a() usa await para chamar as funções c() e d(), o que indica que elas são funções assíncronas que retornam promessas. No entanto, como mencionado anteriormente, essas funções não têm código assíncrono real, o que torna o uso de await desnecessário.