1 .Call async from non-async

We have a "regular" function called f. How can you call the async function wait() and use its result inside of f?

```
async function wait() {
  await new Promise(resolve => setTimeout(resolve, 1000));

  return 10;
}

function f() {
  // ...what should you write here?
  // we need to call async wait() and wait to get 10
  // remember, we can't use "await"
}
```

2 . Rewrite 'if' into '?'

Rewrite this if using the conditional operator '?':

```
let result;

if (a + b < 4) {
  result = 'Below';
} else {
  result = 'Over';
}
```

3. Replace "for" with "while"

Rewrite the code changing the `for` loop to `while` without altering its behavior (the output should stay same).

```
for (let i = 0; i < 3; i++) {
  alert( `number ${i}!` );
}
```

4. You are building a simple React project that fetches data from a server. The server responds with a list of objects representing people's movie reviews. Each object has the following structure:

```
[
  {
    name: "Amit",
    rating: 8, // Rating out of 10
    liked: true // Whether they liked the movie
  },
  {
    name: "Mei",
    rating: 6,
    liked: false
  },
  {
    name: "Priya",
    rating: 9,
    liked: true
  },
  {
    name: "Wei",
    rating: 7,
    liked: true
  }
]
```

Write a JavaScript function that:

1. **Filters** the reviews to include only those where liked is true.
2. **Maps** the filtered list to an array of names of people who liked the movie.
3. **Calculates** the average rating of the filtered reviews using the reduce method.

Provide your solution as **a JavaScript function** that takes an array of review objects as input and returns an object with:

A list of names of people who liked the movie.
The average rating of their reviews.

Expected output:

```
{
  likedBy: ["Amit", "Priya", "Wei"],
  averageRating: 8
}
```