Async/Await Assignment

1). How does async/await help with performance and scalability?
2). Is it possible to use async/await with promise chains? If yes, how can this be achieved?
3). Give 3 real world examples where async/await has been used?

4).

```
async function inc(x) {
 x = x + await 1
 return x;
}

async function increment(x){
 x = x + 1
 return x
}

inc(1).then(function(x){
  increment(x).then(function(x){
  console.log(x)
 })
})
```

Find output.

5).

```
let p = new Promise(function (resolve, reject) {
reject(new Error("some error"));
setTimeout(function(){
reject(new Error("some error"));
```

```
},1000)
reject(new Error("some error"));
});
p.then(null, function (err) {
console.log(1);
console.log(err);
}).catch(function (err) {
console.log(2);
console.log(err);
});
```

Find output.

6).

```
async function f1() {
console.log(1);
}
async function f1() {
console.log(2);
}
console.log(3);
f1();
console.log(1);
f2();
async function f2() {
console.log("Go!");
}
```

Find output.

7).

```
function resolveAfterNSeconds(n,x) {
return new Promise(resolve => {
```

```
    setTimeout( ( ) = {
      resolve(x);
    }, n);
  });

 }


 (function(){

  let a = resolveAfterNSeconds(1000,1)
  a.then(async function(x){
    let y = await resolveAfterNSeconds(2000,2)
    let z = await resolveAfterNSeconds(1000,3)
    let p = resolveAfterNSeconds(2000,4)
    let q = resolveAfterNSeconds(1000,5)
    console.log(x+y+z+await p +await q);
  })

})()
```

Find output.

8). Is it possible to nest async functions in JavaScript? Explain with examples.
9). What is the best way to avoid deadlocks when using async/await?
10). In which scenarios would you use synchronous code instead of asynchronous code?