Question 1:

```
console.log('Start');

setTimeout(() => {
  console.log('Timeout 1');
  Promise.resolve().then(() => console.log('Promise inside Timeout 1'));
}, 0);

Promise.resolve().then(() => {
  console.log('Promise 1');
  setTimeout(() => console.log('Timeout inside Promise 1'), 0);
  return Promise.resolve('Promise 2');
}).then((res) => {
  console.log(res);
});

setTimeout(() => console.log('Timeout 2'), 0);

console.log('End');
```

Output:

Question 2:

```
async function foo() {
  console.log(1);
  await bar();
  console.log(2);
}

async function bar() {
  console.log(3);
  setTimeout(() => console.log(4), 0);
  return Promise.resolve().then(() => console.log(5));
}


foo();


setTimeout(() => console.log(6), 0);


console.log(7);
```

Output:

Question 3:

```
console.log('A');


setTimeout(() => {
  console.log('B');
}, 100);


Promise.resolve().then(() => {
  console.log('C');
  setTimeout(() => {
    console.log('D');
  }, 50);
  Promise.resolve().then(() => {
    console.log('E');
  });
});


setTimeout(() => {
  console.log('F');
  Promise.resolve().then(() => {
    console.log('G');
  });
}, 0);


console.log('H');
```

Output:

Question 4:

```
async function async1() {
  console.log("async1 start");
  await async2();
  console.log("async1 end");
}

async function async2() {
  console.log("async2 start");
  return new Promise((resolve) => {
    console.log("async2 promise start");
    resolve();
    console.log("async2 promise end");
  });
}

console.log("script start");

setTimeout(() => {
  console.log("setTimeout");
}, 0);

async1();

new Promise((resolve) => {
  console.log("promise1");
  resolve();
}).then(() => {
  console.log("promise2");
});

console.log("script end");
```

Output:

Question 5:

```
const task = (time, msg) => {
  return new Promise((resolve) => {
    setTimeout(() => {
      console.log(msg);
      resolve();
    }, time);
  });
};

console.log('Start');

task(1000, 'Timeout 1')
  .then(() => {
    console.log('Promise 1');
    return task(500, 'Timeout 2');
  })
  .then(() => {
    console.log('Promise 2');
  });

setTimeout(() => console.log('Timeout 3'), 500);

Promise.resolve()
  .then(() => {
    console.log('Promise 3');
    setTimeout(() => console.log('Timeout 4'), 0);
  })
  .then(() => console.log('Promise 4'));

console.log('End');
```

Output:

Question 6:

```
console.log('A');

setTimeout(() => {
  console.log('B');
  Promise.resolve().then(() => {
    console.log('C');
  });
}, 100);

setTimeout(() => {
  console.log('D');
}, 0);

Promise.resolve().then(() => {
  console.log('E');
});

Promise.resolve().then(() => {
  setTimeout(() => {
    console.log('F');
  }, 0);
  return Promise.resolve();
}).then(() => {
  console.log('G');
});

console.log('H');
```

Output:

Question 7:

```
async function async1() {
  console.log("async1 start");
  await async2();
  console.log("async1 end");
}

async function async2() {
  console.log("async2 start");
  setTimeout(() => {
    console.log("async2 setTimeout");
  }, 50);
  return Promise.resolve().then(() => {
    console.log("async2 promise");
  });
}

console.log("script start");

setTimeout(() => {
  console.log("setTimeout");
}, 0);

async1();

new Promise((resolve) => {
  console.log("promise1");
  resolve();
}).then(() => {
  console.log("promise2");
});

console.log("script end");
```

Output:

Question 8:

```
console.log('Start');

async function test() {
  console.log('Inside test');
  return Promise.resolve().then(() => console.log('Promise in test'));
}

async function test2() {
  await test();
  console.log('After test');
}

test2().then(() => {
  console.log('Test2 done');
});

console.log('End');
```

Output:

Question 9:

```
console.log(1);

setTimeout(() => {
  console.log(2);
}, 100);

Promise.resolve().then(() => {
  console.log(3);
  return Promise.resolve().then(() => {
    console.log(4);
  });
});

setTimeout(() => {
  console.log(5);
}, 0);

console.log(6);
```

Output:

Question 10:

```
console.log('A');

setTimeout(() => {
  console.log('B');
}, 0);

Promise.resolve().then(() => {
  console.log('C');
  return Promise.resolve().then(() => {
    console.log('D');
  }).then(() => {
    console.log('E');
  });
});

setTimeout(() => {
  console.log('F');
}, 0);

console.log('G');
```

Output:

Question 11:

```
async function async1() {
  console.log(1);
  await async2();
  console.log(2);
}

async function async2() {
  console.log(3);
  return Promise.resolve().then(() => console.log(4));
}

async1();

setTimeout(() => {
  console.log(5);
}, 0);

Promise.resolve().then(() => console.log(6));

console.log(7);
```

Output:

Question 12:

```
setTimeout(() => {
  console.log('Timeout 1');
  Promise.resolve().then(() => {
    console.log('Promise inside Timeout 1');
  });
}, 100);

Promise.resolve().then(() => {
  console.log('Promise 1');
  setTimeout(() => console.log('Timeout inside Promise 1'), 0);
  return Promise.resolve().then(() => {
    console.log('Promise inside Promise 1');
  });
});

setTimeout(() => console.log('Timeout 2'), 0);

console.log('End');
```

Output:

Question 13:

```
console.log('Start');

setTimeout(() => {
  console.log('Timeout 1');
}, 0);

Promise.resolve().then(() => {
  console.log('Promise 1');
  setTimeout(() => {
    console.log('Timeout inside Promise 1');
  }, 0);
  return Promise.resolve();
}).then(() => {
  console.log('Promise 2');
});

setTimeout(() => {
  console.log('Timeout 2');
}, 100);

console.log('End');
```

Output:

Question 14:

```
const asyncFunction = async () => {
  console.log(1);
  await new Promise((resolve) => setTimeout(resolve, 0));
  console.log(2);
};

console.log(3);

asyncFunction().then(() => console.log(4));

setTimeout(() => console.log(5), 0);

console.log(6);
```

Output:

Question 15:

```
console.log("Start");

setTimeout(() => {
  console.log("Timeout 1");
  Promise.resolve().then(() => {
    console.log("Promise after Timeout 1");
  });
}, 0);

Promise.resolve().then(() => {
  console.log("Promise 1");
}).then(() => {
  console.log("Promise 2");
});

console.log("End");

setTimeout(() => {
  console.log("Timeout 2");
}, 0);
```

Output: