Q1. What are conditional statements? Explain conditional statements with syntax and examples.

Conditional statements are used to perform different actions based on different conditions. These statements allow a program to make decisions and execute code blocks depending on whether a condition is true or false. The most common conditional statements are if, else if, and else.

Syntax:

```
if (condition) {
   // code block to be executed if condition is true
} else if (another_condition) {
   // code block to be executed if another_condition is true
} else {
   // code block to be executed if neither condition is true
}
```

Example

```
let age = 18;
if (age >= 18) {
   console.log("You are an adult.");
} else {
   console.log("You are a minor.");
}
```

In this example, the program checks if the age is greater than or equal to 18. If true, it prints "You are an adult.", otherwise, it prints "You are a minor."

Q2. Write a program that grades students based on their marks.

```
function gradeStudent(marks) {
   if (marks > 90) {
      return "A Grade";
   } else if (marks >= 70 && marks <= 90) {
      return "B Grade";
   } else if (marks >= 50 && marks < 70) {
      return "C Grade";
   } else {
      return "F Grade";
```

```
  }
}


let marks = 85;
console.log(gradeStudent(marks));  // Output: B Grade
```

This function takes a student's marks as input and returns the grade based on the specified conditions:

- If marks are greater than 90, they get an "A" grade.
- If marks are between 70 and 90, they get a "B" grade.
- If marks are between 50 and 70, they get a "C" grade.
- If marks are below 50, they get an "F" grade.

Q3. What are loops, and what do we need them? Explain different types of loops with their syntax and examples.

Loops are used to repeatedly execute a block of code while a condition is true. They help avoid repetition and make the code more efficient by running a piece of code multiple times.

There are several types of loops:

1. **For loop**
   A for loop runs a block of code a set number of times.

Syntax:

javascript

Copy

```
for (initialization; condition; increment) {
  // code to be executed
}
```

Example:

javascript

Copy

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

This loop will print numbers 0 to 4.

2. **While loop**
   A while loop runs as long as the condition is true.

Syntax:

javascript

Copy

```
while (condition) {
  // code to be executed
}
```

Example:

javascript

Copy

```
let i = 0;
while (i < 5) {
  console.log(i);
  i++;
}
```

This loop will also print numbers 0 to 4.

3. **Do-while loop**
   A do-while loop guarantees that the code block will run at least once before checking the condition.

Syntax:

javascript

Copy

```
do {
  // code to be executed
} while (condition);
```

Example:

javascript

Copy

```
let i = 0;
do {
  console.log(i);
```

```
    i++;
} while (i < 5);
```

This loop will print numbers 0 to 4 as well, but the condition is checked after the loop runs.

Q4. Generate numbers between any 2 given numbers.

You can generate numbers between two given values using a loop.

javascript

Copy

```
const num1 = 10;
const num2 = 25;

for (let i = num1 + 1; i <= num2; i++) {
    console.log(i);
}
```

Output:

Copy

```
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
```

This code starts from num1 + 1 and goes up to num2, printing each number in between.

Q5. Use the while loop to print numbers from 1 to 25 in ascending and descending order.

Ascending order (1 to 25):

javascript

Copy

```
let i = 1;
while (i <= 25) {
   console.log(i);
   i++;
}
```

Descending order (25 to 1):

javascript

Copy

```
let i = 25;
while (i >= 1) {
   console.log(i);
   i--;
}
```

This code will print numbers from 1 to 25 in ascending order and 25 to 1 in descending order.