Conditions and Loops Assignment Questions

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

Ans:- Conditional statements in JavaScript are used to make decisions in code by executing different blocks of code based on whether a specified condition evaluates to true or false. These statements help control the flow of a program and allow it to respond to different situations.

There are primarily three types of conditional statements in JavaScript:

1. **if statement:** The if statement is used to execute a block of code if a specified condition is true.
2. if (condition) {
3.   // Code to be executed if the condition is true

}

Example:

```
let x = 10;


if (x > 5) {
  console.log("x is greater than 5");
}
```

4. **if-else statement:** The if-else statement is used to execute one block of code if a specified condition is true and another block of code if the condition is false.

5. if (condition) {
6.   // Code to be executed if the condition is true
7. } else {
8.   // Code to be executed if the condition is false

}

Example:

```
let y = 3;

if (y % 2 === 0) {
  console.log("y is even");
} else {
  console.log("y is odd");
}
```

9. **if-else if-else statement:** The if-else if-else statement is an extension of the if-else statement, allowing you to check multiple conditions in a sequence.
10. if (condition1) {
11.   // Code to be executed if condition1 is true
12. } else if (condition2) {
13.   // Code to be executed if condition2 is true
14. } else {
15.   // Code to be executed if none of the conditions is true

}

Example:

```
let grade = 75;
```

```
if (grade >= 90) {
  console.log("A");
} else if (grade >= 80) {
  console.log("B");
} else if (grade >= 70) {
  console.log("C");
} else {
  console.log("F");
}
```

These conditional statements are fundamental for building logic and controlling the flow of your JavaScript programs. Depending on the conditions, different code blocks will be executed, allowing for flexible and dynamic behavior in your applications.

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

8 If greater than 90 then A GradE

8 If between 70 and 90 then a B gradE

8 If between 50 and 70 then a C gradE

8 Below 50 then an F grade.

Ans:- const studentMarks = 49;

```
if(studentMarks >= 90){
   console.log('Grade A');
}else if(studentMarks >= 70 && studentMarks < 90){
   console.log('Grade B');
}else if(studentMarks >= 50 && studentMarks < 70){
   console.log('Grade C');
}else{
   console.log('Grade F');
}
```

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

Ans:- Loops in JavaScript are used to execute a block of code repeatedly as long as a specified condition is true. They are essential for automating repetitive tasks, iterating over data structures, and performing operations on a set of values. JavaScript provides several types of loops, each serving different purposes.

1. for loop:

The for loop is used when the number of iterations is known or when iterating over a range of values.

```
for (initialization; condition; increment/decrement) {
  // Code to be executed in each iteration
}
```

Example:

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

2. while loop:

The while loop is used when the number of iterations is not known beforehand, and the loop continues as long as a specified condition is true.

```
while (condition) {
  // Code to be executed in each iteration
}
```

Example:

```
let i = 0;


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

3. do-while loop:

The do-while loop is similar to the while loop, but it guarantees that the code block is executed at least once before checking the condition.

```
do {
  // Code to be executed in each iteration
} while (condition);
```

Example:

```
let i = 0;


do {
  console.log(i);
  i++;
} while (i < 5);
```

4. for...in loop:

The for...in loop is used to iterate over the properties of an object.

```
for (variable in object) {
  // Code to be executed in each iteration
}
```

Example:

```
const person = {
  name: 'John',
  age: 30,
  job: 'Developer'
};

for (let key in person) {
  console.log(key + ': ' + person[key]);
}
```

5. for...of loop:

The for...of loop is used to iterate over the values of an iterable object (e.g., arrays, strings).

```
for (variable of iterable) {
  // Code to be executed in each iteration
}
```

Example:

```
const colors = ['red', 'green', 'blue'];

for (let color of colors) {
  console.log(color);
}
```

Loops are essential in programming because they allow you to execute a block of code repeatedly without duplicating it. They are particularly useful when working with arrays, lists, and other data structures, enabling efficient processing of elements and reducing code redundancy.

Q4. Generate numbers between any 2 given numbers.

Ex 8 const num1 = 10

8 const num2 = 25;

Output: 11, 12, 13, ..., 25

Ans:- const num1 = 10;

```
const num2 = 25;


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

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

Ans:- const num1 = 1;

```
const num2 = 25;



console.log(`Ascending number from ${num1} to ${num2} are:`);
for(let start = num1; start <= num2; start++){
   console.log(start);
}


console.log("\n");
console.log(`Descending number from ${num2} to ${num1} are:`);
for(let start = num2; start >= num1; start--){
    console.log(start);
}
```

Complete