Certainly! Let's break down **Question 5: Loops in C** in detail.

---

5. Loops in C

Loops are used to **execute a block of code multiple times** until a condition is met. C provides three main types of loops:

1. **for loop**
2. **while loop**
3. **do-while loop**

Each loop type has its own use case, depending on the situation.

---

1. for Loop

The `for` loop is used when the number of iterations is **known beforehand**.

Syntax:

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

- **Initialization**: Variable is initialized (e.g., `int i = 0`).
- **Condition**: The loop runs while the condition is `true`.
- **Increment/Decrement**: Updates the loop variable (`i++` or `i--`).

Example: for Loop

```
#include <stdio.h>
int main() {
    for (int i = 1; i <= 5; i++) {
        printf("Iteration %d\n", i);
    }
    return 0;
}
```

Output:

```
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
```

---

2. while Loop

The `while` loop is used when **the number of iterations is unknown**, and it continues **until a condition becomes false**.

Syntax:
```
while (condition) {
    // Code executes while condition is true
}
```

Example: while Loop
```
#include <stdio.h>
int main() {
    int count = 1;
    while (count <= 5) {
        printf("Count: %d\n", count);
        count++;  // Incrementing count
    }
    return 0;
}
```

Output:

```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

---

3. do-while Loop

The `do-while` loop is similar to the `while` loop, but it **executes at least once**, even if the condition is `false`.

Syntax:
```
do {
    // Code executes at least once
} while (condition);
```

Example: do-while Loop
```
#include <stdio.h>
int main() {
    int num = 1;
    do {
        printf("Number: %d\n", num);
        num++;
    } while (num <= 5);
    return 0;
}
```

Output:

```
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
```

Special Case: do-while Executes Even if Condition is False

```c
#include <stdio.h>
int main() {
    int num = 10;
    do {
        printf("Executed at least once!\n");
    } while (num < 5); // Condition is false, but runs once
    return 0;
}
```

Output:

```
Executed at least once!
```

---

4. Nested Loops

A loop inside another loop is called a **nested loop**.

Example: Nested for Loop (Printing a Pattern)

```c
#include <stdio.h>
int main() {
    for (int i = 1; i <= 3; i++) {  // Outer Loop
        for (int j = 1; j <= 3; j++) {  // Inner Loop
            printf("* ");
        }
        printf("\n"); // New line after each row
    }
    return 0;
}
```

Output:

```
* * *
* * *
* * *
```

---

5. Infinite Loops

If the condition never becomes `false`, the loop will **run forever** (infinite loop).

Example: Infinite while Loop

```c
#include <stdio.h>
int main() {
    while (1) {  // Always true
        printf("This loop runs forever!\n");
    }
    return 0;
}
```

(Press `Ctrl + C` to stop the program manually.)

---

6. Using break and continue in Loops

(a) break Statement

The `break` statement **exits** the loop immediately.

Example: Using break

```c
#include <stdio.h>
int main() {
    for (int i = 1; i <= 10; i++) {
        if (i == 5) {
            break;  // Exits when i == 5
        }
        printf("%d\n", i);
    }
    return 0;
}
```

Output:

```
1
2
3
4
```

(The loop **stops at 4** because `i == 5` breaks it.)

(b) continue Statement

The `continue` statement **skips** the current iteration and moves to the next one.

Example: Using continue

```c
#include <stdio.h>
int main() {
    for (int i = 1; i <= 5; i++) {
        if (i == 3) {
            continue;  // Skips when i == 3
        }
        printf("%d\n", i);
    }
    return 0;
}
```

Output:

```
1
2
4
5
```

(Loop **skips** `3` but continues running.)

Key Differences Between Loops

Loop Type	When to Use	Executes At Least Once?
`for` loop	When number of iterations is **known**	No
`while` loop	When number of iterations is **unknown**	No
`do-while` loop	When code must run **at least once**	Yes

Key Takeaways

✓ **for loop**: Best when the number of iterations is **known**.
✓ **while loop**: Used when the number of iterations is **unknown**.
✓ **do-while loop**: Executes **at least once**, even if the condition is false.
✓ **break**: Immediately **exits** the loop.
✓ **continue**: **Skips** the current iteration and moves to the next.
✓ **Nested loops**: Loops inside loops for multi-dimensional structures.

Would you like any additional examples or explanations?