7. Loop Control Statements

• THEORY EXERCISE: Explain the use of break, continue, and goto statements in C. Provide examples of each.

1. break Statement

- The **break** statement is used to terminate the nearest enclosing loop (**for**, **while**, or **do-while**) or a **switch** statement. When a **break** statement is encountered, control is transferred to the statement immediately following the loop or **switch**.

```
#include <stdio.h>

int main() {
   for (int i = 0; i < 10; i++) {
      if (i == 5) {
         break; // Exit the loop when i is 5
      }
      printf("%d\n", i);
}
   printf("Loop terminated at i = 5\n");
}
```

2. continue Statement

- The **continue** statement is used to skip the current iteration of a loop and proceed to the next iteration. In a **for** loop, it jumps to the increment/decrement step, while in a **while** or **do-while** loop, it jumps to the condition check.

Example:

```
#include <stdio.h>


int main() {
   for (int i = 0; i < 10; i++) {
      if (i % 2 == 0) {
         continue; // Skip the even numbers
      }
```

```
        printf("%d\n", i);
    }
}
```

3. goto Statement

- The **goto** statement is used to transfer control to a labeled statement within the same function. It can lead to less readable and maintainable code, so its use is generally discouraged in favor of structured control flow constructs like loops and conditionals.

Example:

```
#include <stdio.h>


int main() {
    int i = 0;


    loop_start:
    if (i < 5) {
        printf("%d\n", i);
        i++;
        goto loop_start; // Jump back to the labeled statement
    }
}
```

- **break**: Exits the nearest enclosing loop or **switch** statement. Use it to terminate loops based on a condition.
- **continue**: Skips the current iteration of a loop and proceeds to the next iteration. Use it to filter out specific cases within loops.
- **goto**: Transfers control to a labeled statement. Use it cautiously, as it can lead to unstructured and hard-to-read code.