Looping in C

7.Compare and contrast while loops, for loops, and do-while loops. Explainthescenarios in which each loop is most.

→ **while Loop:**

→ **Structure:** Checks the condition before executing the loop body.

→ **Syntax:**

→ while (condition) {

→ // Loop body

→ }

→ **Usage:** Best when the number of iterations is not known beforehand. It continues as long as the condition is true.

→ **Example Scenario:** Reading input until a sentinel value is encountered (e.g., reading numbers until a user enters zero).

→ **for Loop:**

→ **Structure:** Combines initialization, condition checking, and increment/decrement in a single line.

→ **Syntax:**

→ for (initialization; condition; increment) {

→ // Loop body

→ }

→ **Usage:** Ideal when the number of iterations is known in advance, such as iterating over arrays or fixed ranges.

→ **Example Scenario:** Iterating through the elements of an array or performing a calculation a specific number of times (e.g., calculating factorial).

→ **do-while Loop:**

→ **Structure:** Executes the loop body first, then checks the condition afterward, ensuring at least one execution.

→ **Syntax:**

→ do { // Loop body

→ } while (condition);

→ **Usage:** Useful when the loop body must be executed at least once regardless of the condition.

→ **Example Scenario:** Prompting the user for input and validating that input, ensuring the prompt is displayed at least once.