When to use For loop, while loop and do while loop?

Use for loop when:
Definite Iteration: You know the number of iterations beforehand.

```
for (int i = 1; i <= 10; i++) {
   // Code to be executed in each iteration
}
```

Iterating Over Arrays or Sequences: When working with arrays or sequences, a **for** loop can be more concise.

```
for (int i = 0; i < arraySize; i++) {
   // Code to process elements of an array
}
```

Use while loop when:
Indefinite Iteration: You don't know the number of iterations beforehand, and the loop is controlled by a condition.

```
int input;

while (input != 0) {
   // Code to be executed as long as input is not zero
   printf("Enter a number (0 to exit): ");
   scanf("%d", &input);
}
```

Input Validation: When validating user input or performing a task until a certain condition is met.

```
int number;

printf("Enter a positive number: ");
scanf("%d", &number);
```

```
while (number <= 0) {
   printf("Invalid input. Please enter a positive number: ");
   scanf("%d", &number);
}
```

Use do-while loop when:

- **Guaranteed Execution at Least Once:** You want to ensure that a block of code is executed at least once, regardless of the loop condition.

```
int guess;
int secretNumber = 7;

do {
   printf("Guess the number (1 to 10): ");
   scanf("%d", &guess);

   if (guess == secretNumber) {
      printf("Congratulations! You guessed the correct number.\n");
   } else {
      printf("Incorrect guess. Try again.\n");
   }

} while (guess != secretNumber);
```

Menu-Driven Programs: In menu-driven programs, where the menu is displayed and the user is expected to make a choice.

```
char choice;

do {
   // Display menu options
```

```
    printf("Do you want to continue? (y/n): ");
    scanf(" %c", &choice);

} while (choice == 'y' || choice == 'Y');
```

In many cases, the choice between these loops is a matter of personal preference and code readability. It's important to choose the loop that best fits the logic of your program and makes your code clear and maintainable.