**Exercise 1: Vector Operations**

Create a numeric vector with elements from 1 to 10. Perform the following operations:

- ★ Calculate the sum of all elements in the vector.
- ★ Replace all even numbers with their squares.
- ★ Print all elements greater than 5.
- ★ Calculate the mean of all elements in the vector.
- ★ Count and print how many elements are multiples of 3.

**Exercise 2: Vector Manipulation**

Create a numeric vector with elements from 10 to 1. Perform the following operations:

- ★ Print all elements of the vector in reverse order.
- ★ Replace all elements less than 5 with 0.
- ★ Calculate the product of all elements greater than 5.
- ★ Print elements that are odd numbers.
- ★ Calculate the cumulative sum of the vector elements.

**Exercise 3: Conditional Vector Handling**

Create a numeric vector with elements from 1 to 20. Perform the following operations:

★ Print elements that are divisible by 4.
★ Replace all multiples of 3 with -1.
★ Print elements that are perfect squares (e.g., 1, 4, 9, 16).
★ Calculate the median of the vector elements.
★ Print elements that are prime numbers.

**Exercise 4: Vector Filtering and Transformation**

Create a numeric vector with elements from 1 to 15. Perform the following operations:

★ Print elements that are multiples of both 2 and 3.
★ Replace all elements less than 5 with 0 and greater than 10 with 10.
★ Calculate the range (difference between maximum and minimum) of the vector.
★ Print elements that are powers of 2 (e.g., 1, 2, 4, 8).
★ Calculate the sum of all elements divisible by 5.

**Exercise 5: Advanced Vector Operations**

Create a numeric vector with elements from 1 to 25. Perform the following operations:

★ Print elements that are even numbers.
★ Replace all elements divisible by 3 with their square roots (if they are perfect squares).

★ Calculate the standard deviation of the vector elements.
★ Print elements that are Fibonacci numbers.
★ Calculate the product of all prime numbers in the ve**ctor.**

## Exercise-6: Student Grade Analysis

1. Create a vector called `student_scores` with the following exam scores for 15 students:

78, 85, 92, 67, 88, 79, 93, 95, 70, 82, 86, 89, 75, 91, 84

2. Calculate and print the average score for the class.

3. Find and print the highest and lowest scores.

4. Create a vector `grade_boundaries` with the following values:

60, 70, 80, 90

5. Use a series of if-else statements to assign letter grades to each score:

- Below 60: F

- 60-69: D

- 70-79: C

- 80-89: B

- 90 and above: A

Store these grades in a new vector called `letter_grades`.

6. Count how many students received each letter grade and print the results.

7. Identify any scores that are more than 10 points below the class average. Print these scores and their positions in the `student_scores` vector.

8. Calculate the median score of the class.

9. Create a new vector `normalized_scores` that adjusts all scores by adding 5 points, but caps the maximum score at 100.

10. Use a for loop to print each student's original score, normalized score, and letter grade in the format:

"Student X: Original Score - [score], Normalized Score - [norm_score], Grade - [grade]"

11. Determine if the class performed better in the first half or the second half of the exam. Assume the first 7 scores represent the first half and the last 8 scores represent the second half. Print which half had a higher average score.

12. Create a vector `pass_fail` where scores of 70 and above are marked as "Pass" and below 70 are marked as "Fail". Use vector operations, not a loop.

13. Calculate and print the percentage of students who passed the exam.

**Exercise 7: E-commerce Sales Analysis**

1. Create a vector `daily_sales` with 30 days of sales data (in dollars):

120, 145, 130, 180, 190, 210, 160, 150, 170, 140, 200, 220, 190, 180, 170,

155, 165, 175, 185, 195, 205, 215, 225, 235, 245, 255, 265, 275, 285, 295

2. Calculate and print the total sales for the month.

3. Find the day with the highest sales and the day with the lowest sales. Print both the day numbers and the sales amounts.

4. Calculate the average daily sales for the month.

5. Create a vector `weekend_sales` containing only the sales from weekends (assume the first two elements are a weekend, then every 5th and 6th element after that).

6. Compare the average weekend sales to the average weekday sales. Print which is higher and by how much.

7. Create a vector `sales_categories` where sales are categorized as follows:

- "Low" for sales under 150
- "Medium" for sales between 150 and 200
- "High" for sales 200 and above

Use vector operations, not a loop.

8. Count how many days fall into each sales category and print the results.

9. Calculate the cumulative sales for each day (running total). Store this in a new vector `cumulative_sales`.

10. Use a for loop to print the day number and cumulative sales for every 5th day.

11. Determine if there was any streak of 5 or more consecutive days where sales were above the monthly average. Print the result.

12. Create a vector `sales_change` that shows the day-to-day change in sales. The first element should be 0.

13. Use a while loop to find the first day when the cumulative sales exceeded $3000. Print this day number.

## Exercise 8: Population Growth Simulation

1. Create a vector `initial_population` with 5 values representing different species in an ecosystem:

   100, 50, 200, 80, 30

2. Create a vector `growth_rates` with annual growth rates for each species:

   1.1, 1.15, 1.05, 1.08, 1.2

3. Simulate population growth for 10 years. Use a for loop to calculate the population of each species each year. Store the results in a matrix `population_matrix` where each row represents a year and each column a species.

4. Calculate and print the total population of all species combined for each year.

5. Identify which species has the largest population after 10 years. Print both the species number and its population.

6. Create a vector `endangered` that is TRUE for any species with a population under 50 in the final year, and FALSE otherwise.

7. Use a for loop to print the years when any species first exceeded 500 individuals. If a species never exceeded 500, print a message saying so.

8. Calculate the percentage change in population for each species from year 1 to year 10. Store these in a vector `percent_change`.

9. Create a vector `population_categories` where species are categorized based on their final population:

   - "Small" for populations under 100

- "Medium" for populations between 100 and 500
- "Large" for populations over 500

10. Use a while loop to determine how many years it takes for the total population of all species combined to double from the initial total. Print this number of years.

11. Calculate the average population for each species across all years. Store these in a vector `avg_population`.

12. Create a logical vector `stable_population` that is TRUE for species whose population in the final year is within 10% of their average population, and FALSE otherwise.