Algorithm Assignment - 1

Similar to previous assignments, you should start by initializing a new Git repository and adhering to best practices. For this assignment, you are required to implement ten algorithms. Each one should be written in a separate file named "`Algo-X.java`", where X represents the algorithm's number. After finalizing each algorithm, please commit the file.

1. Prompt the user to input a number. Ensure it's a positive number. If validated, generate the Fibonacci sequence up to that number using a `do-while` loop.
2. Given the array `int[] nums = {-5, 2, 7, 10, 58, -7, 8, 23}`;, determine the smallest and largest numbers in this set.
3. Given the array `int[] nums = {10, 20, 30, 40, 50};` , Construct a new array and populate it with the elements of the original array, but in the opposite sequence. Then assign the memory location of the new array to `nums` and print out the `nums` array.
4. Using the aforementioned array, invert the order of its elements directly within the same array. You must not create any additional arrays for this task. Once finished, print out the `nums` array.
5. Prompt the user to enter a text (ASCII only). Ensure the input is not empty. If the input is valid, invert the sequence of characters and display the reversed text.
6. Prompt the user to enter a sentence (using ASCII characters). Check to ensure the input isn't empty. If the input passes validation, invert the order of the words and then display the resulting sentence.
7. Prompt the user to provide a sentence. Ensure the input isn't empty. If valid, identify and display the shortest and longest words from the sentence, along with their respective lengths.
8. Given two arrays:

   ```
   int[] numA = {5, 7, -2, 3, 4, 6, 7};
   int[] numB = {7, 8, -8, 2, 1, -9, 6};
   ```

   Determine and print the following:

   1. `numA` ∩ `numB`
   2. `numA` ∪ `numB`
   3. `numA` / `numB`

    4. `numB` / `numA`
    5. `numA` △ `numB`

9. Ask the user to input a single word. If they enter a sentence or multiple words, prompt them again. After receiving a valid word, determine if it's a palindrome. For verification, use the words "noon," "civic," "racecar," and "level." These should be recognized as palindromes.

10. Prompt the user to input a phone number in either of these formats: `+94 XX XXX XXXX` or `0XX-XXXXXXX`, where X represents a digit. To check if a character is a digit, utilize the `Character.isDigit()` method. Do not use any other API for this task. If the phone number aligns with the provided patterns, display "Number is validated." If not, show "Invalid phone number." If the user doesn't provide any input, indicate "Telephone number can't be empty," and request them to enter it again.