**Practical Algorithms and Sums Assignment**

**1.Pascal\_Triangle**

**Step 1:** Create a directory names Algos

****

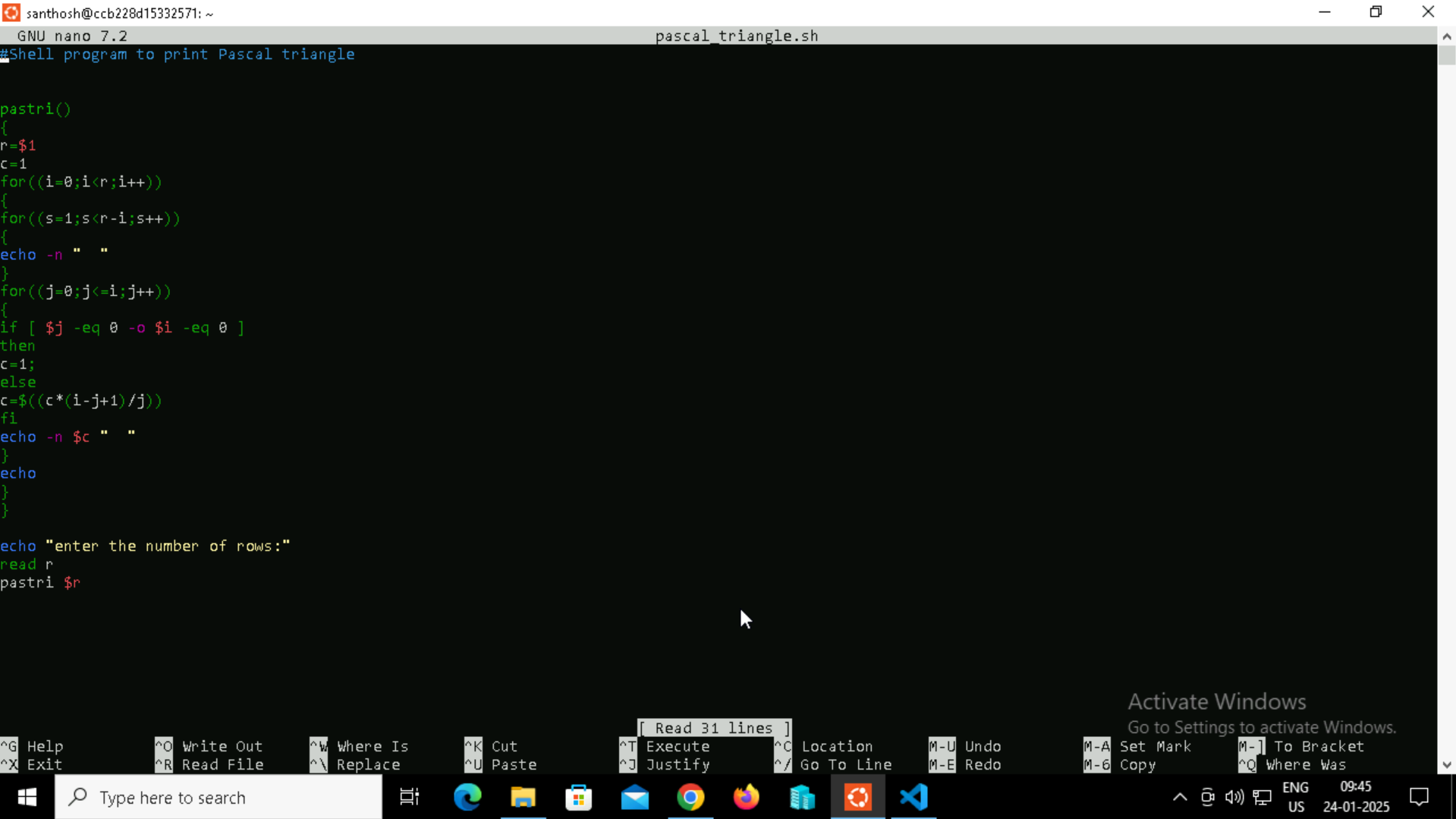
**Step 2**: Editing the saved Pascal\_triangle File using nano command.

****

**Step 3:** Providing the Pascal\_triangle shell script**.**

**Purpose:** The purpose of this shell script is to generate and display a Pascal triangle for a given number of rows.

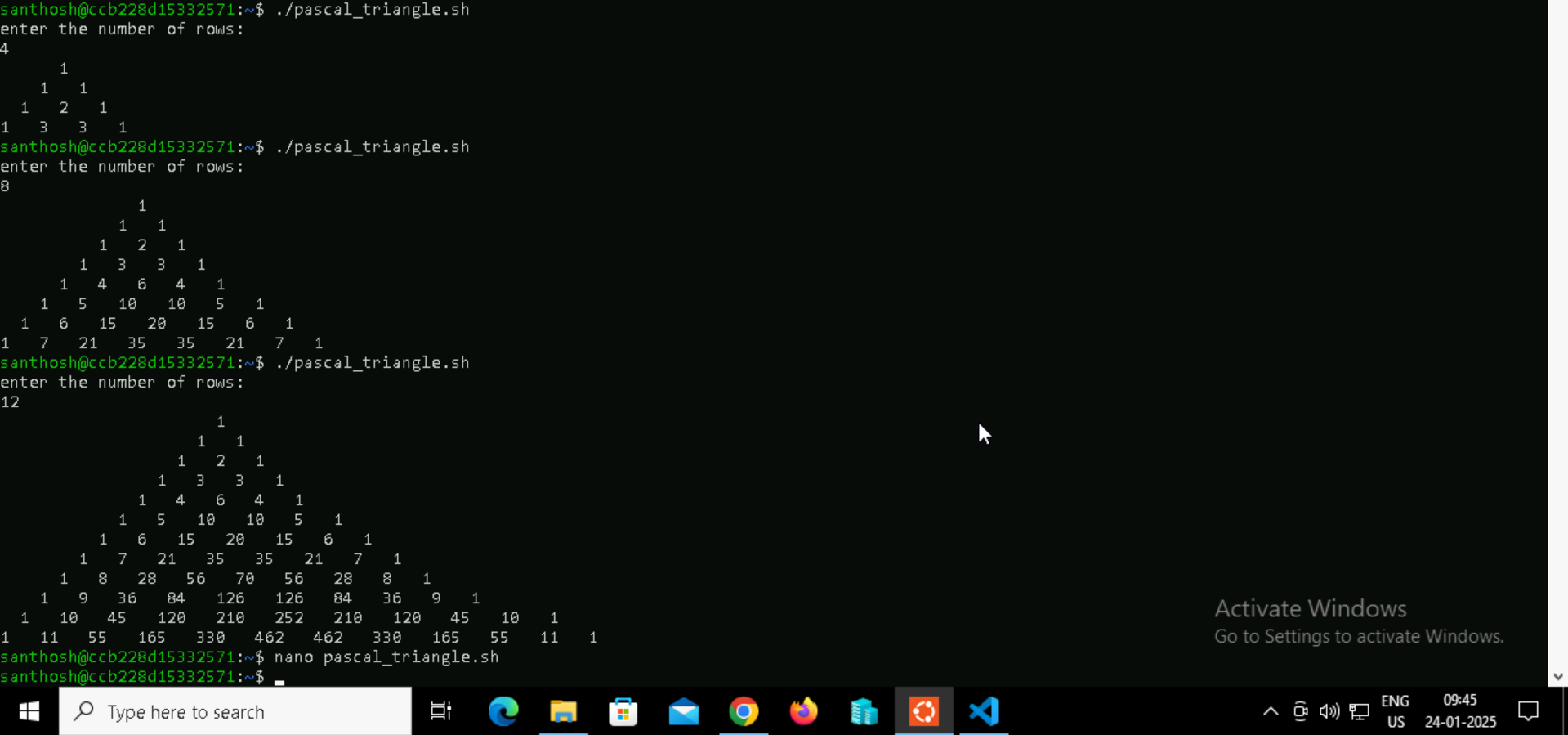
* pastri() defines the Pascal triangle generation function.
* It uses r=$1 to accept the number of rows as an argument.
* The loop for((i=0;i<r;i++)) iterates through the number of rows.
* The loop for((s=1;s<r-i;s++)) ensures proper formatting by adding spaces to center the triangle.
* The loop for((j=0;j<=i;j++)) calculates the binomial coefficients for each position in the row.
* If the current element is the first element of the row, its value is set to 1.
* For other elements, the binomial coefficient is calculated using:
* The computed value is printed using echo -n, which prevents a newline from being added after each value, ensuring they appear on the same row.
* A newline is added at the end of each row using echo.
* The script prompts the user to enter the number of rows using read r.
* The user-provided input is passed as an argument to the pastri function.



**Step 4:** Provide the execute permission to the file.



**Step 5:** Output



**2.Personal Message**

**Step 1:** Touch a file name Personal message and edit it.

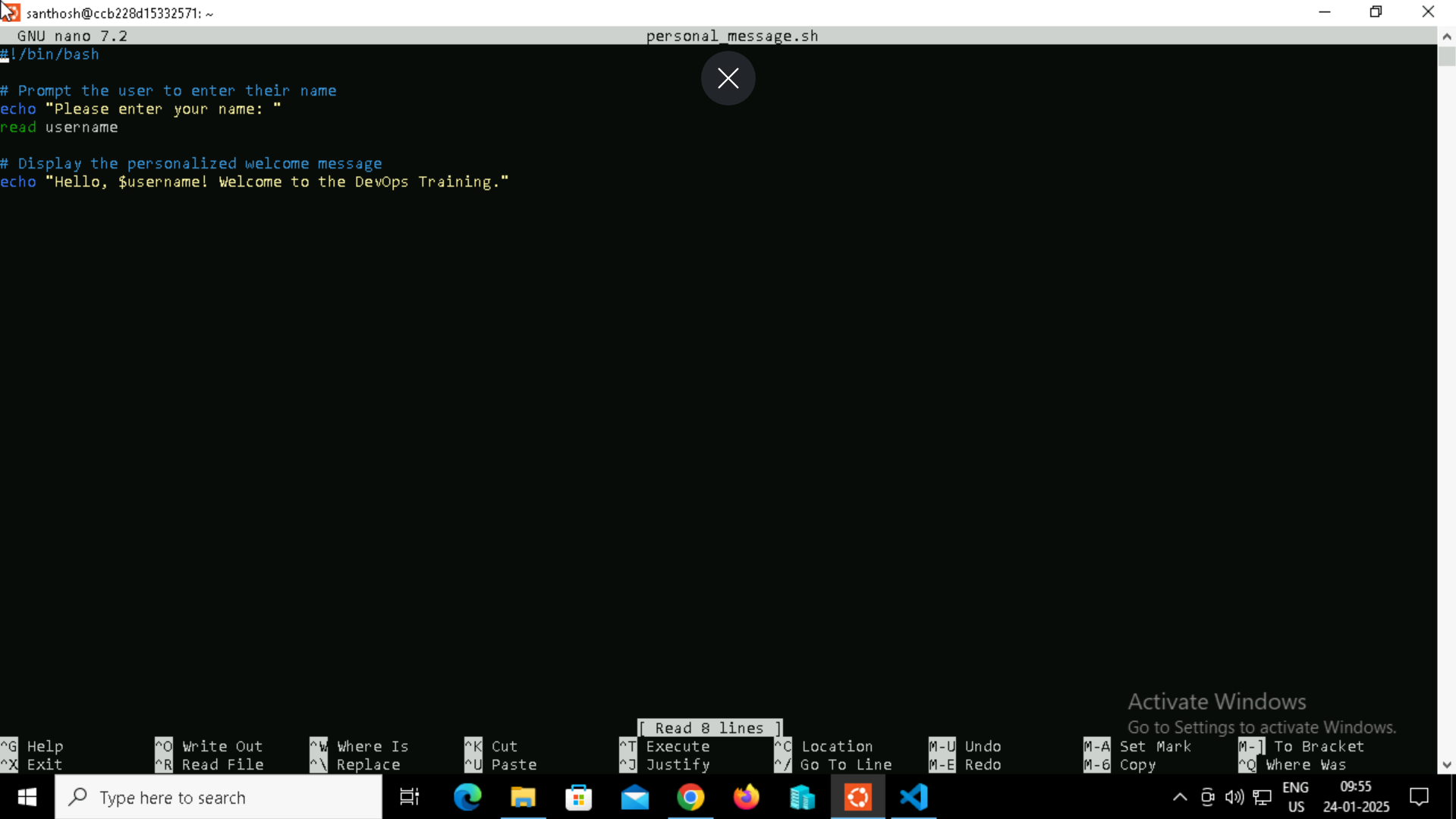
****

**Step 2:** Provide the shell script inside the file.

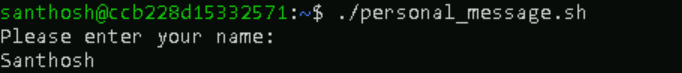
**Purpose:** This document explains a simple shell script that interacts with the user by asking for their name and greeting them with a personalized message.

**Shebang (#!/bin/bash)**:

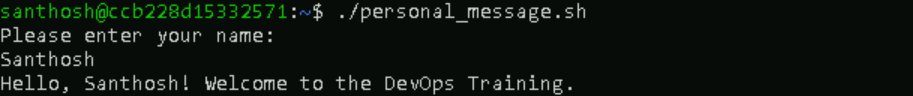
* Specifies that the script should be executed in the bash shell.
* The echo "lease enter your name" command displays a message prompting the user to input their name.
* The read Username command takes the user input and stores it in the variable Username.
* The echo "Hello $Username! Welcome to the DevOps Training." command uses string interpolation to include the user’s input in the greeting message.

****

**Step 3:** Run the shell Script.

****

**Step 4:** Output

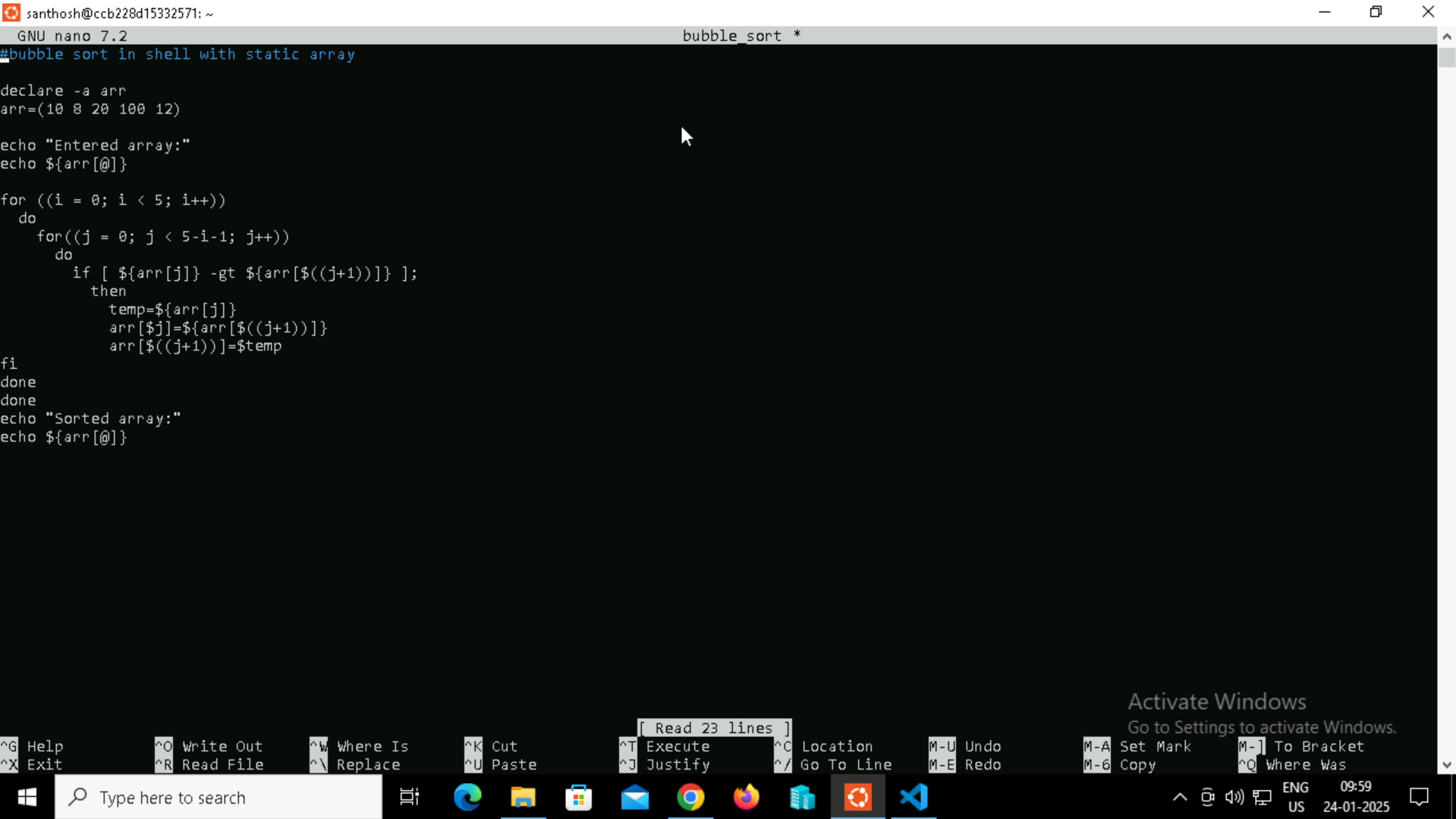
****

**3.Bubble Sort**

**Step 1:** Touch a file name bubble\_sort and edit it



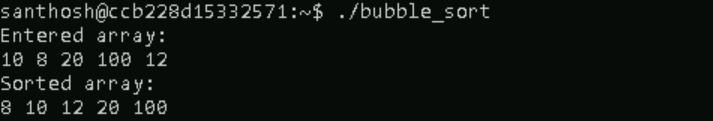
**Step 2:** Provide the shell script inside the file.



**Step 3:** Run the shell Script after providing permissions to execute using chmod.

****

**Step 4:** Output

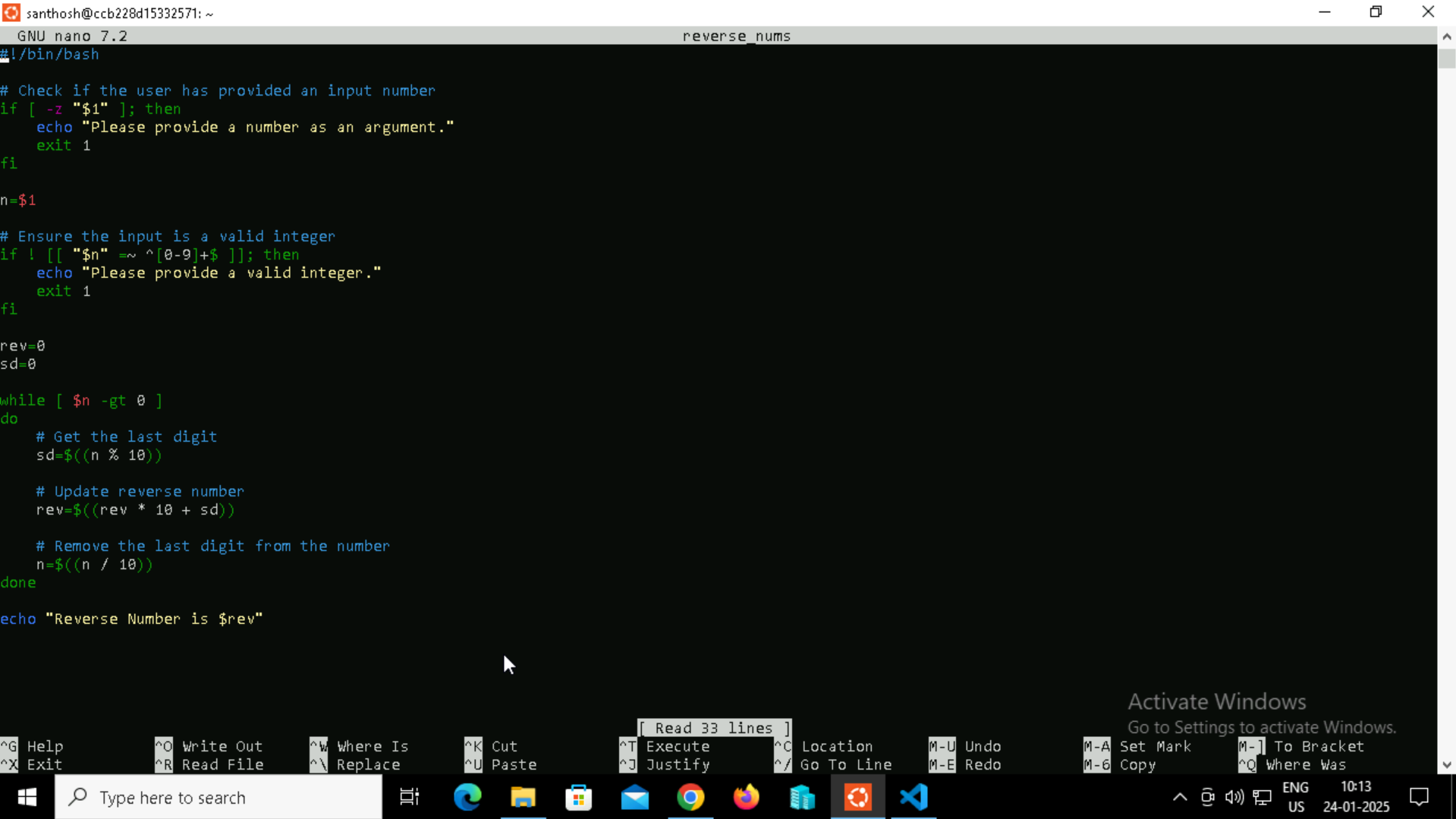
****

**4.Reverse\_nums**

**Step 1:** Touch a file name Reverse\_nums and edit it

****

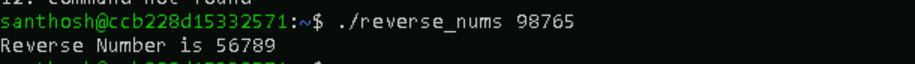
**Step 2:** Provide the shell script inside the file.

****

**Step 3:** Run the shell Script after providing permissions to execute using chmod.



**Step 4:** Output

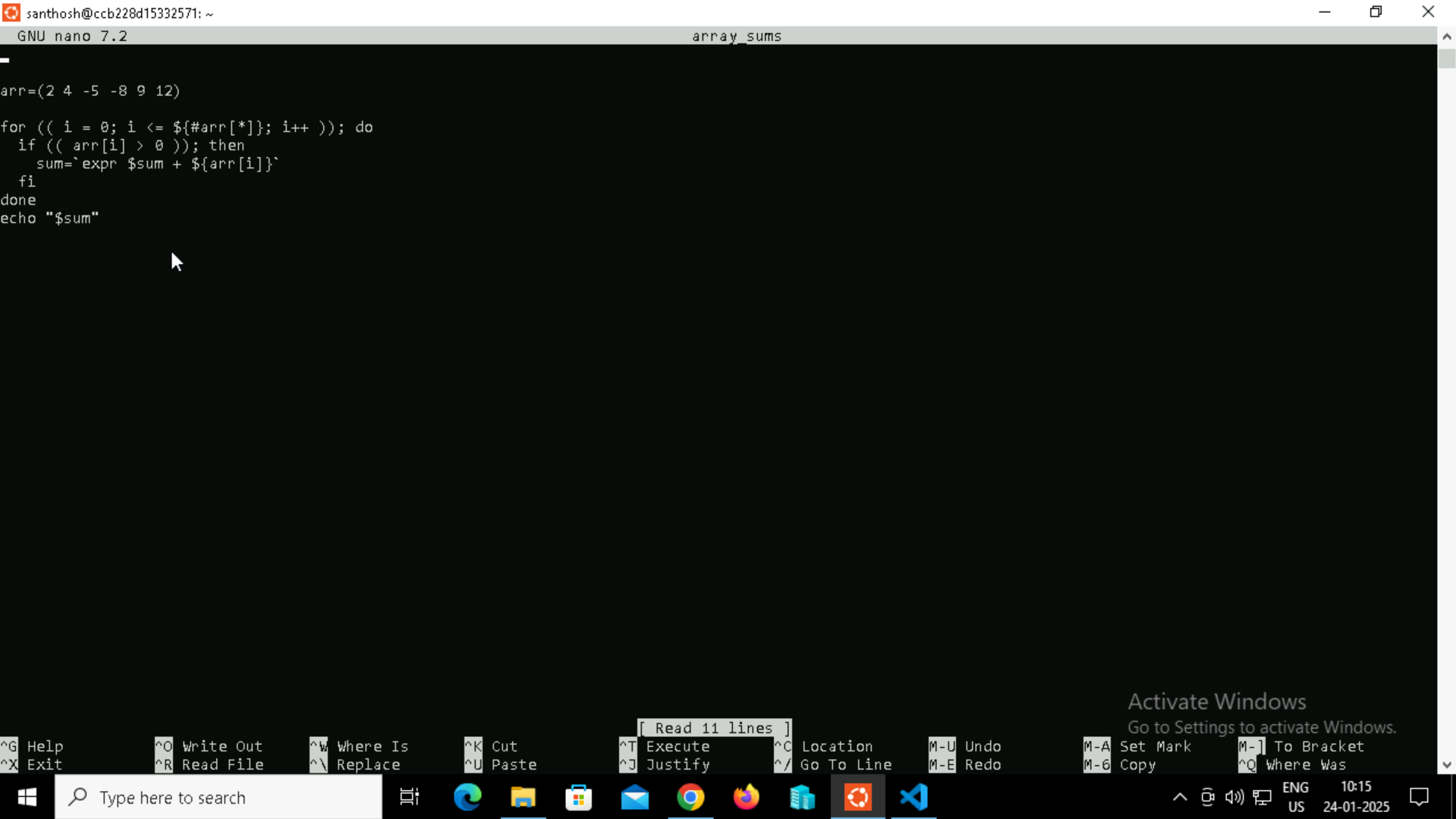


**5.Array\_sums**

**Step 1:** Touch a file name Array\_sums and edit it

****

**Step 2:** Provide the shell script inside the file.



**Step 3:** Run the shell Script after providing permissions to execute using chmod.



**Step 4:** Output

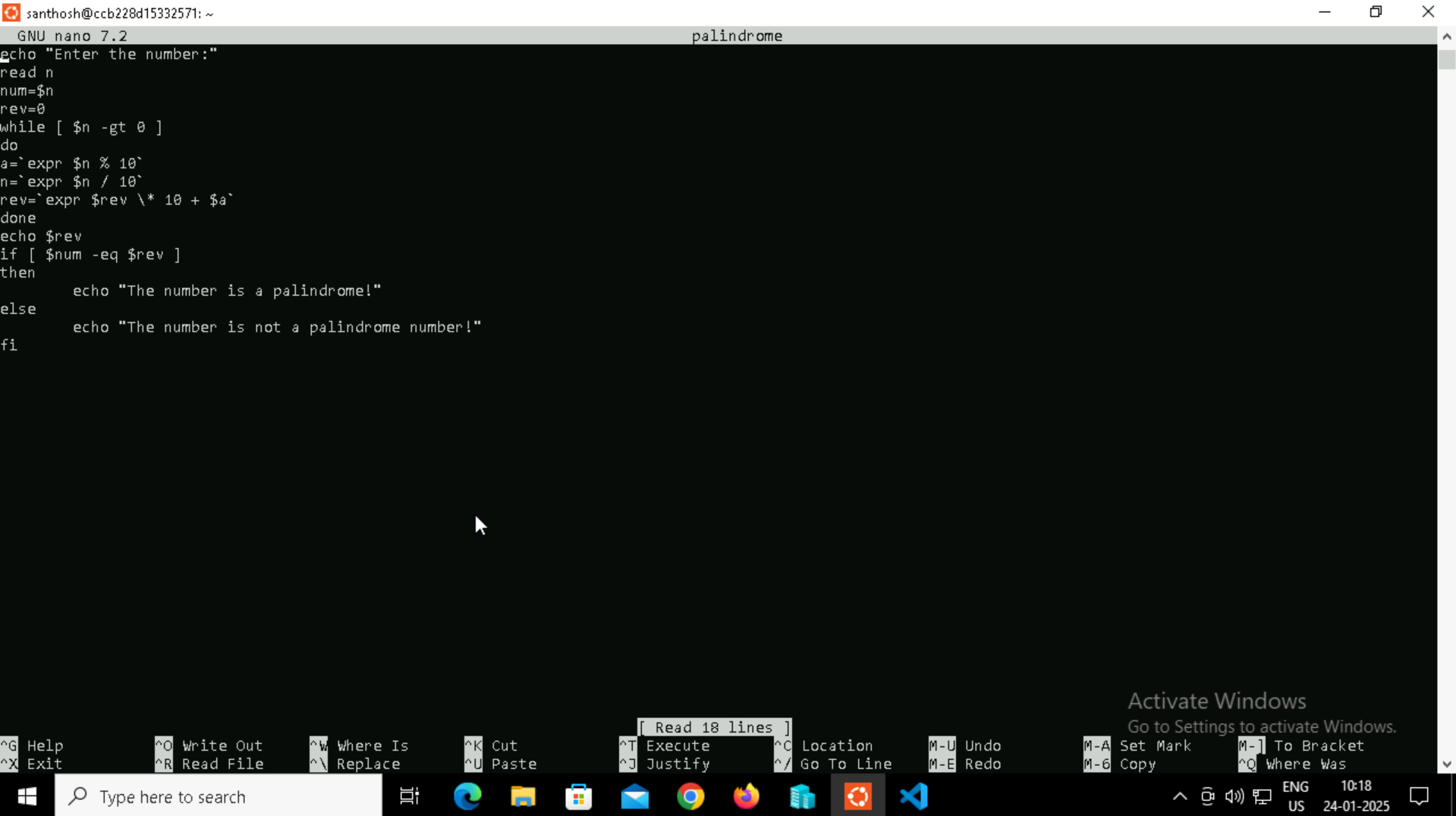


**6.Palindrome**

**Step 1:** Touch a file name Palindrome and edit it

****

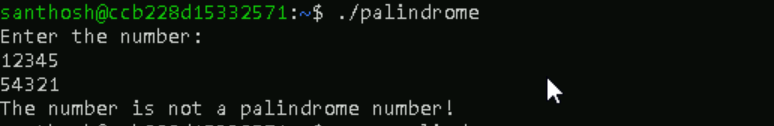
**Step 2:** Provide the shell script inside the file.



**Step 3:** Run the shell Script after providing permissions to execute using chmod.

****

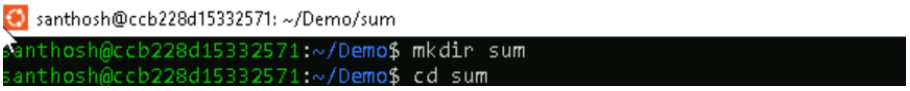
**Step 4:** Output



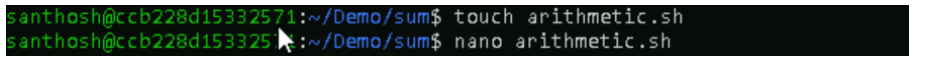
**7. Arithmetic**

**Shell Script: Integer Addition with Input Validation**

**Step 1:** Create a directory first using mkdir sum and change the directory to the sum directory.

****

**Step 2:** Touch and nano a file named arithmetic.sh

****

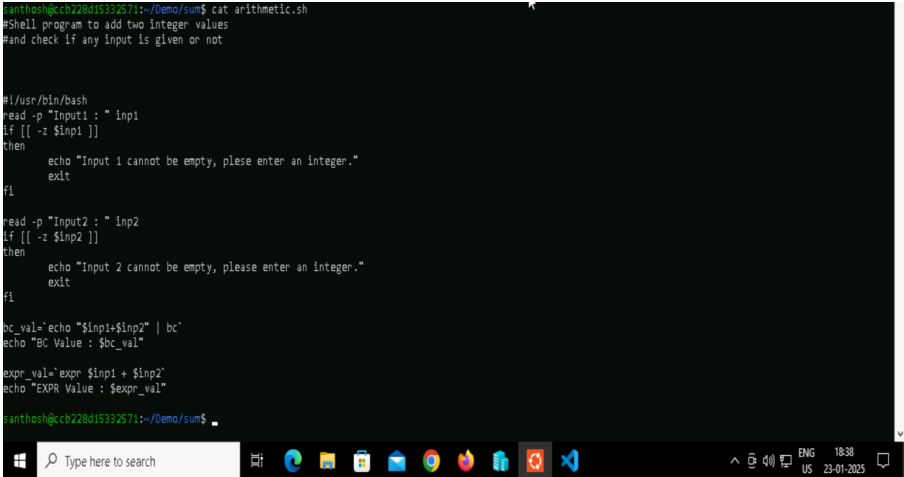
**Step 3:** Code Purpose

1. Prompts the user: Asks the user to enter two integer values.

2. Input Validation: Checks if the user provided input for both values. If either input is empty, an error message is displayed, and the script exits.

3. Calculates Sum: Calculates the sum of the two input integers using the bc command for arbitrary precision arithmetic. Calculates the sum using the expr command for basic integer arithmetic.

4. Displays Results: Prints the calculated sum using both bc and expr.



**Step 4:** Code Explanation • #!/usr/bin/bash specifies that the script should be executed with the Bash interpreter. • read -p "Input1 : " inp1: Prompts the user to enter the first integer and stores it in the inp1 variable. • read -p "Input2 : " inp2: Prompts the user to enter the second integer and stores it in the inp2 variable. • if [[ -z $inp1 ]]: Checks if inp1 is empty. If empty, displays an error message and exits with an error code (1). • if [[ -z $inp2 ]]: Checks if inp2 is empty. If empty, displays an error message and exits with an error code (1). • bc\_val=$(echo "$inp1 + $inp2" | bc): Uses the bc command to calculate the sum with arbitrary precision and stores the result in bc\_val. • expr\_val=$(expr $inp1 + $inp2): Uses the expr command to calculate the sum with basic integer arithmetic and stores the result in expr\_val. • echo "BC Value : $bc\_val": Displays the sum calculated using bc. • echo "EXPR Value : $expr\_val": Displays the sum calculated using expr.

**Step 5:** Running the file using ./arithmetic.sh



**Step 6:** Providing the execut permissions to the arithmetic shell file.



**Step 7:** Output of the shell script.

