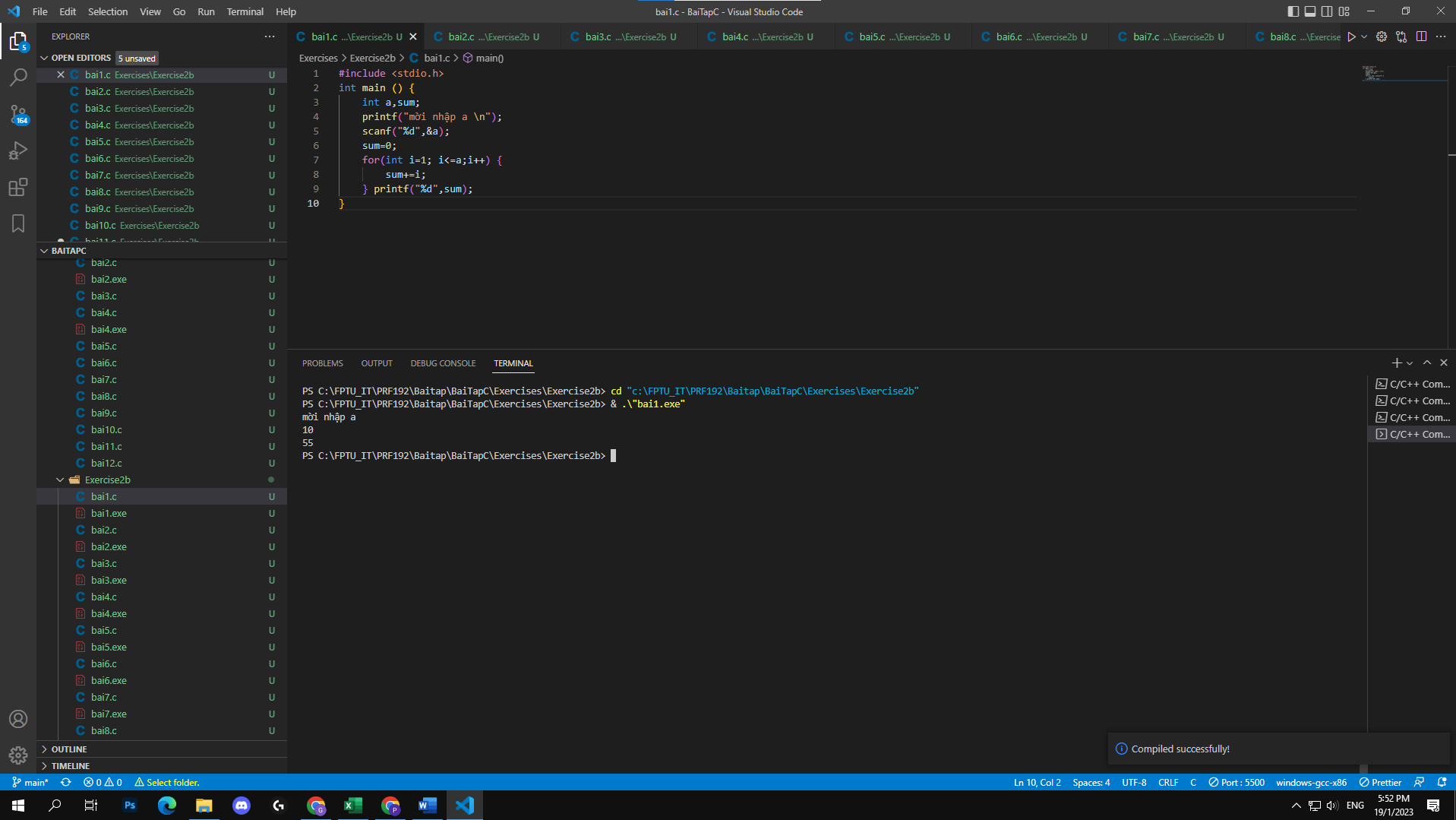
**Practice Exercises - Chapter: 05**

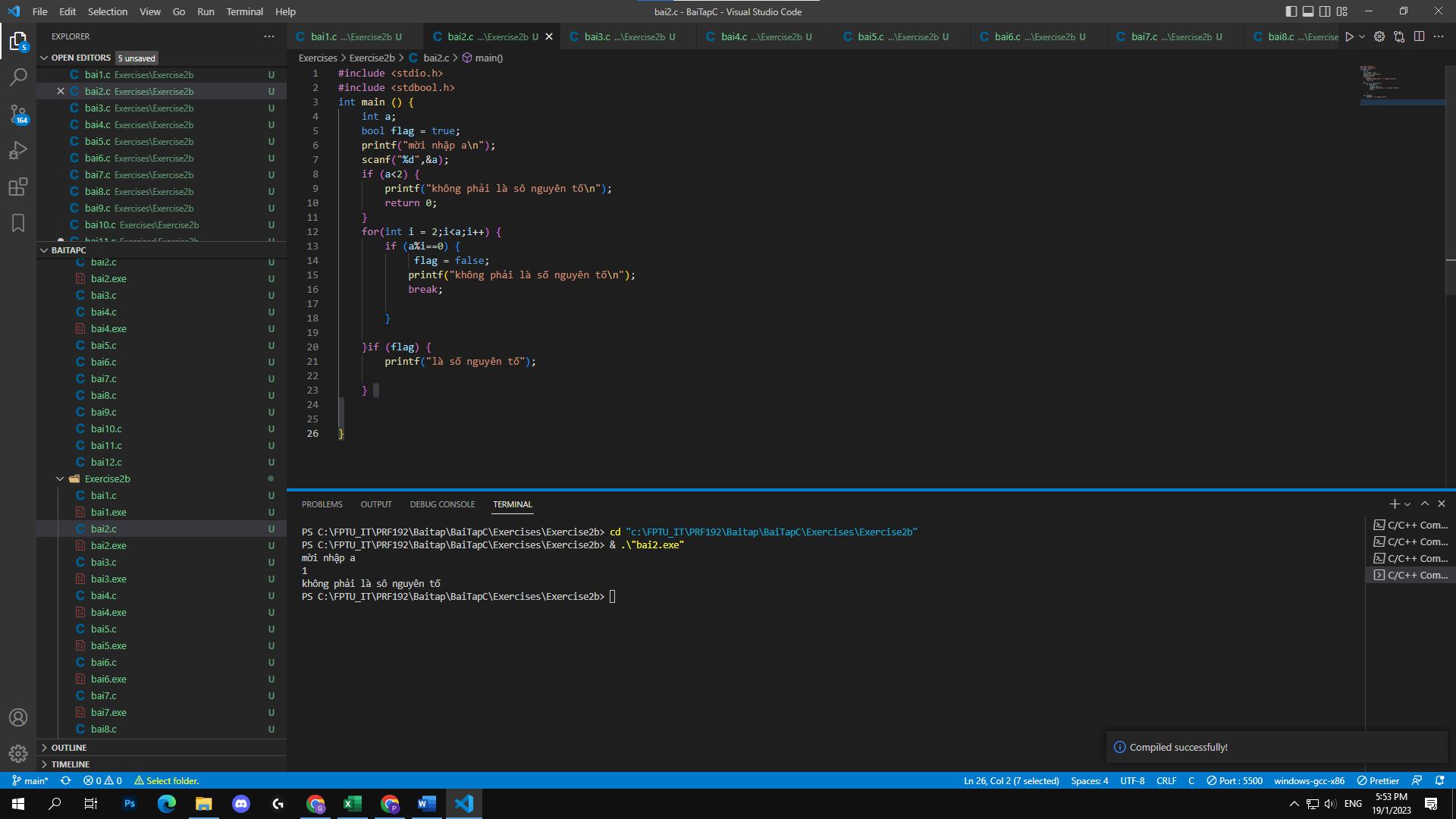
**\* Exercise 5.1: Sum of n numbers**

Write a program to calculate the sum of the numbers from *1* to *n*, where n is entered from the keyboard.



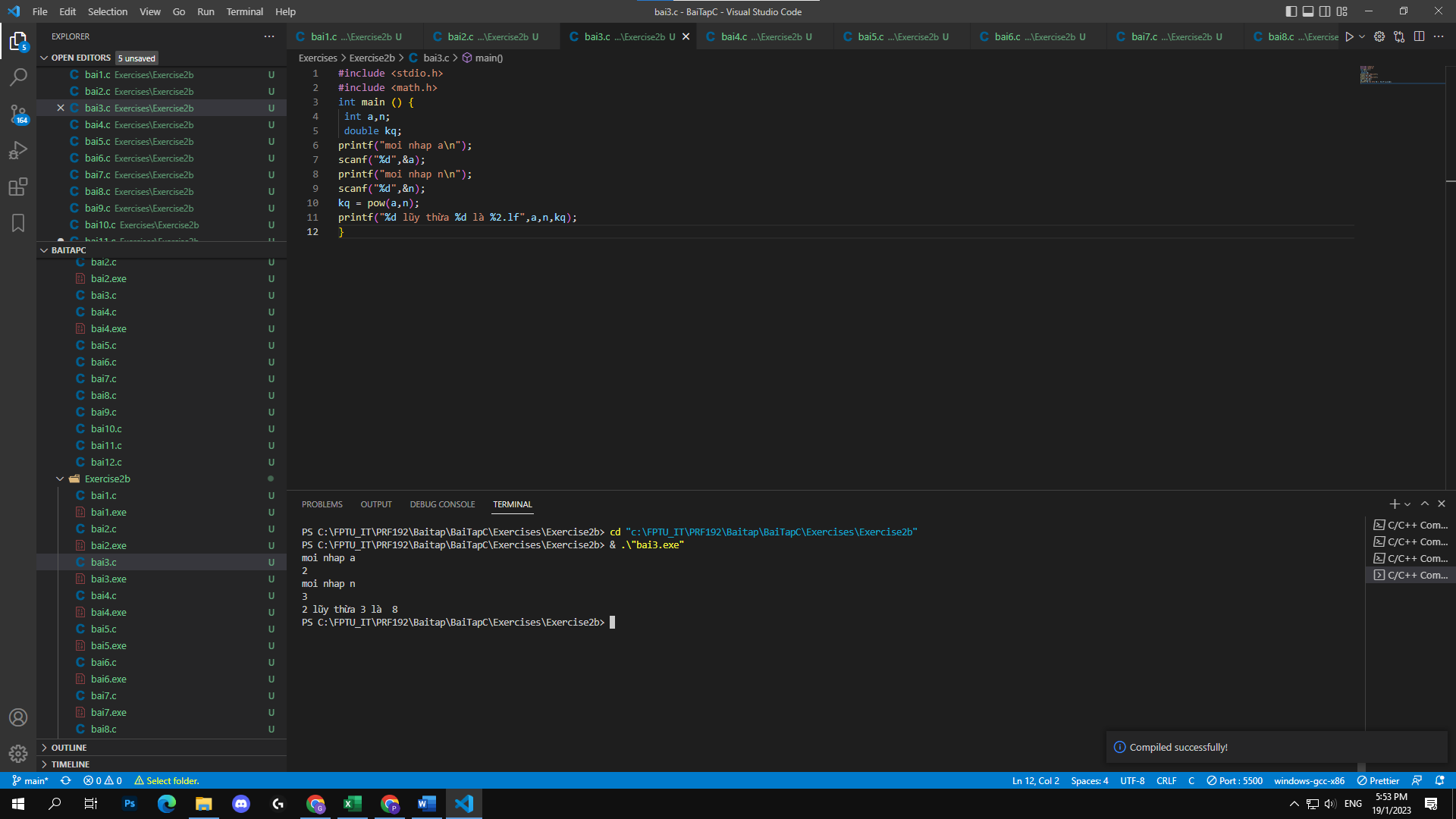
**\* Exercise 5.2: Checking for prime numbers**

Write a program that accepts a number from the keyboard and then check whether the number is prime or not.



**\* Exercise 5.3: Program to calculate power of number**

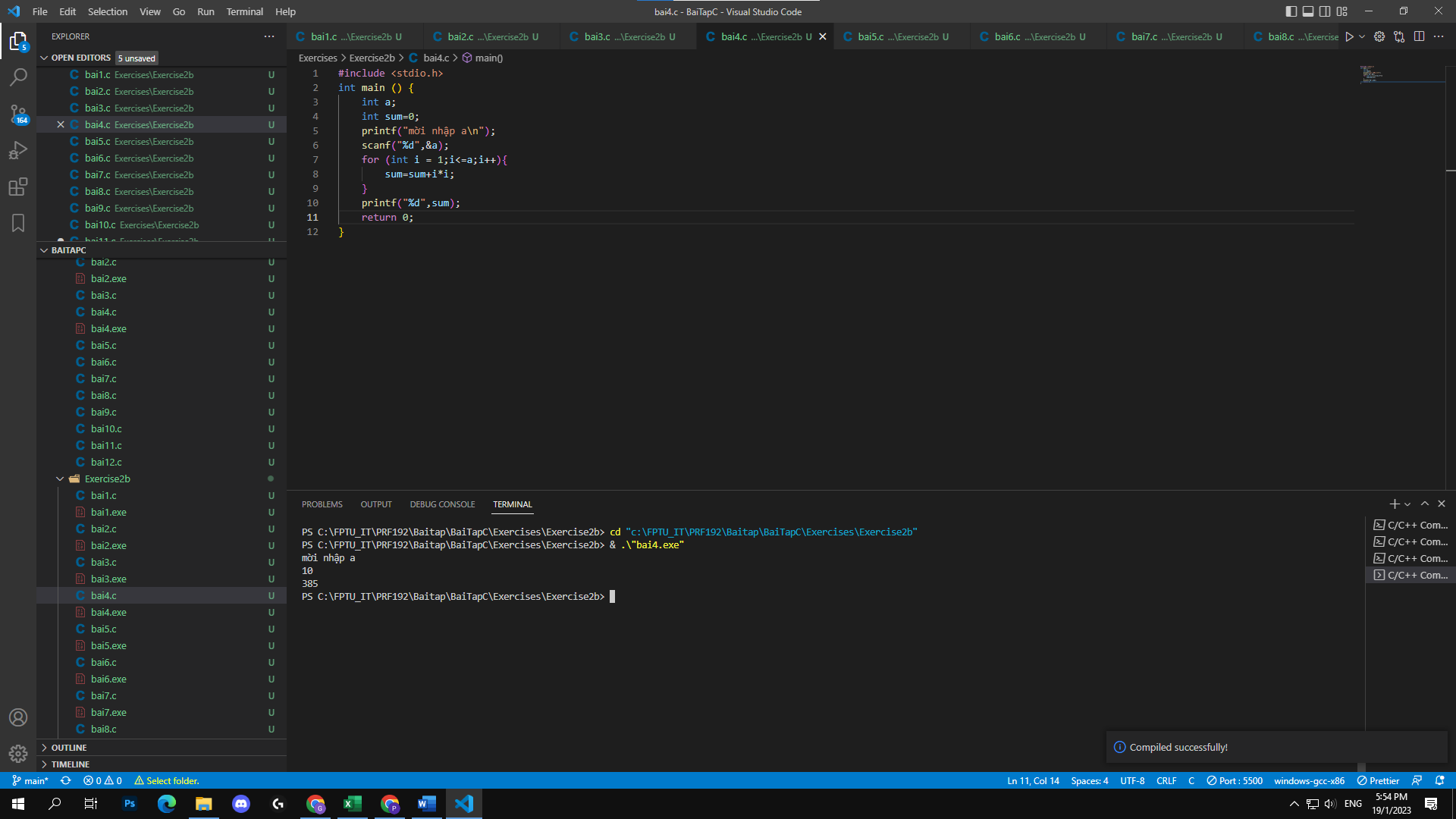
Write a program which takes two numbers a and n from the keyboard. Then calculate and display the value of “a power n” to the screen.



**\* Exercise 5.4: Calculate the value of the expression**

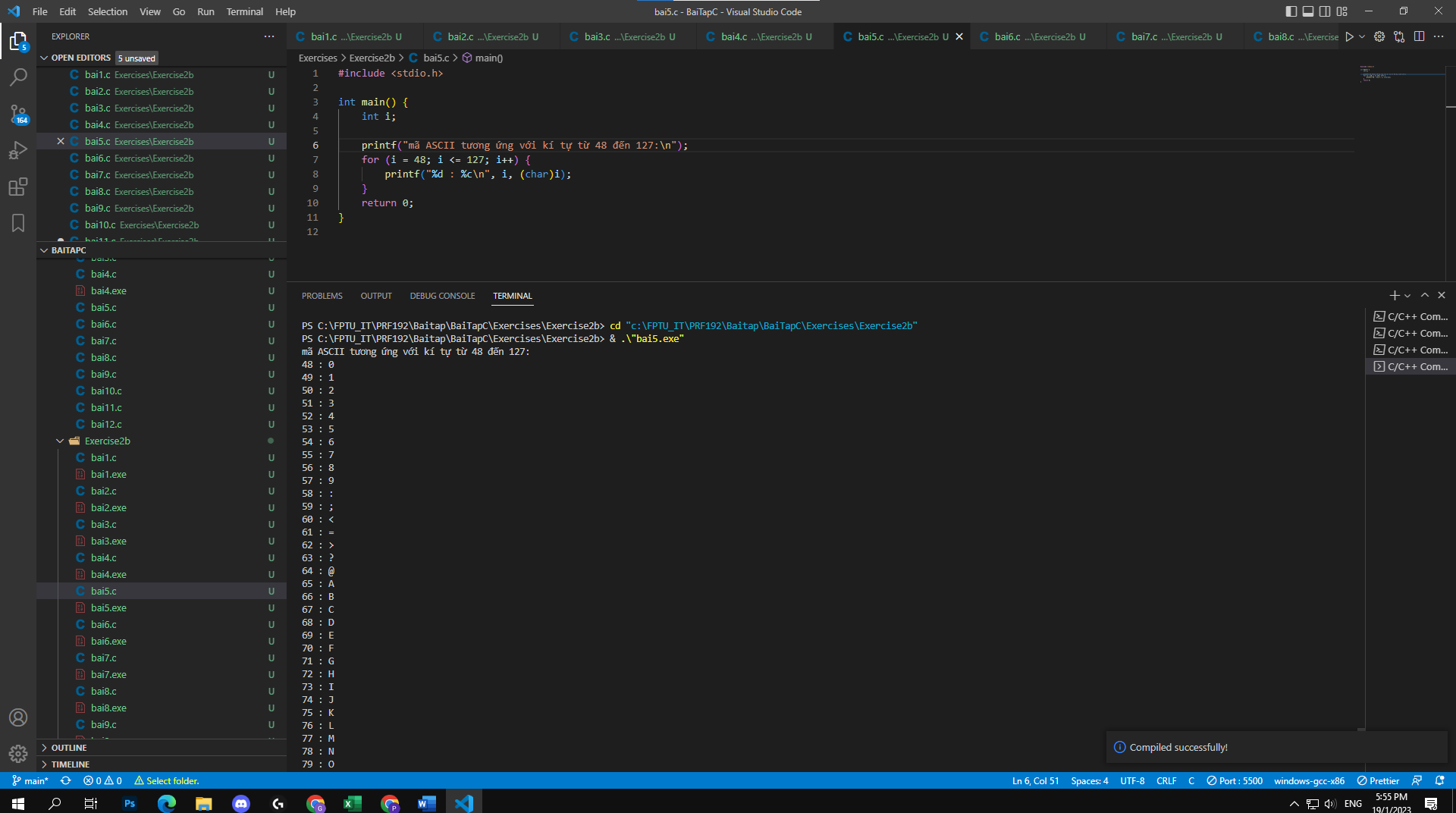
Write a program to input a positive integer *n* from the keyboard, then calculate and display to the screen the value of the following expression:

(1 \* 1) + (2 \* 2) + (3 \* 3) + (4 \* 4) + (5 \* 5) + ... + (n \* n).



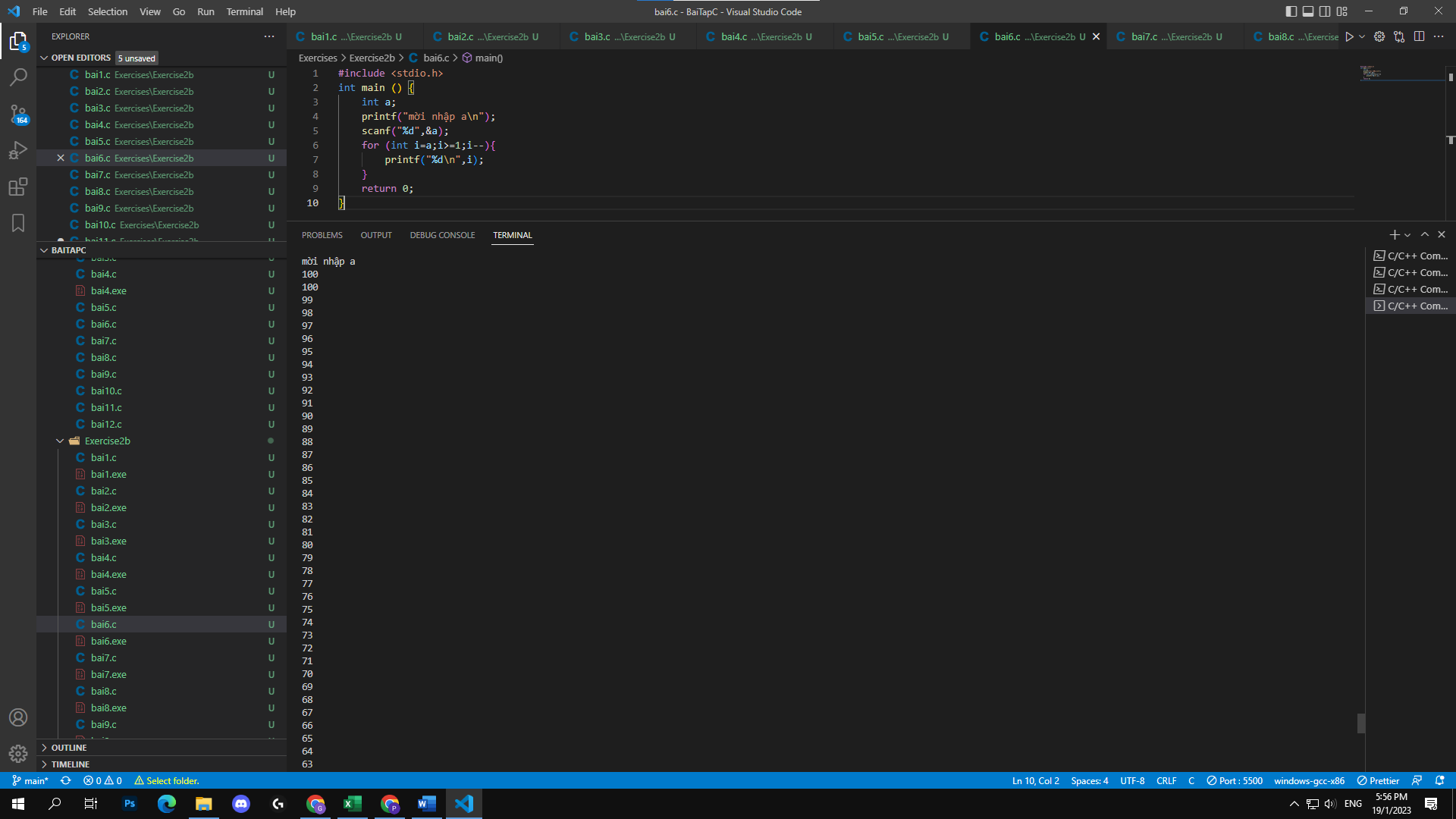
**\* Exercise 5.5: ASCII codes and characters**

Write a program to print all the ASCII codes and their equivalent characters with the ASCII codes from 48 to 127.



**\* Exercise 5.6: Print all natural numbers in reverse order**

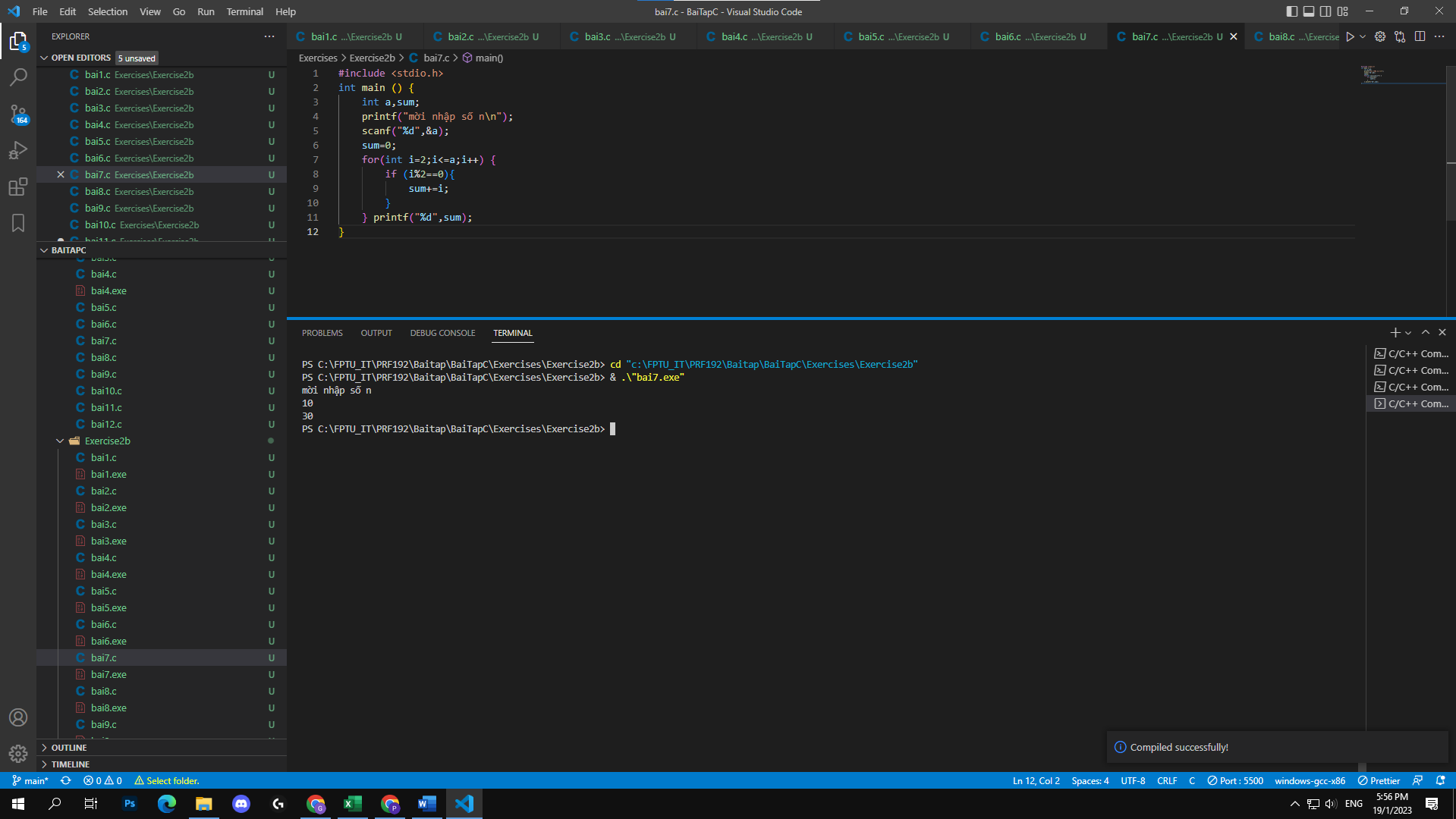
Write a program to input a natural number *n* and display to the screen natural numbers from *n* to *1*. Numbers are separated by a space.



**\* Exercise 5.7: Sum of even numbers from 1 to n**

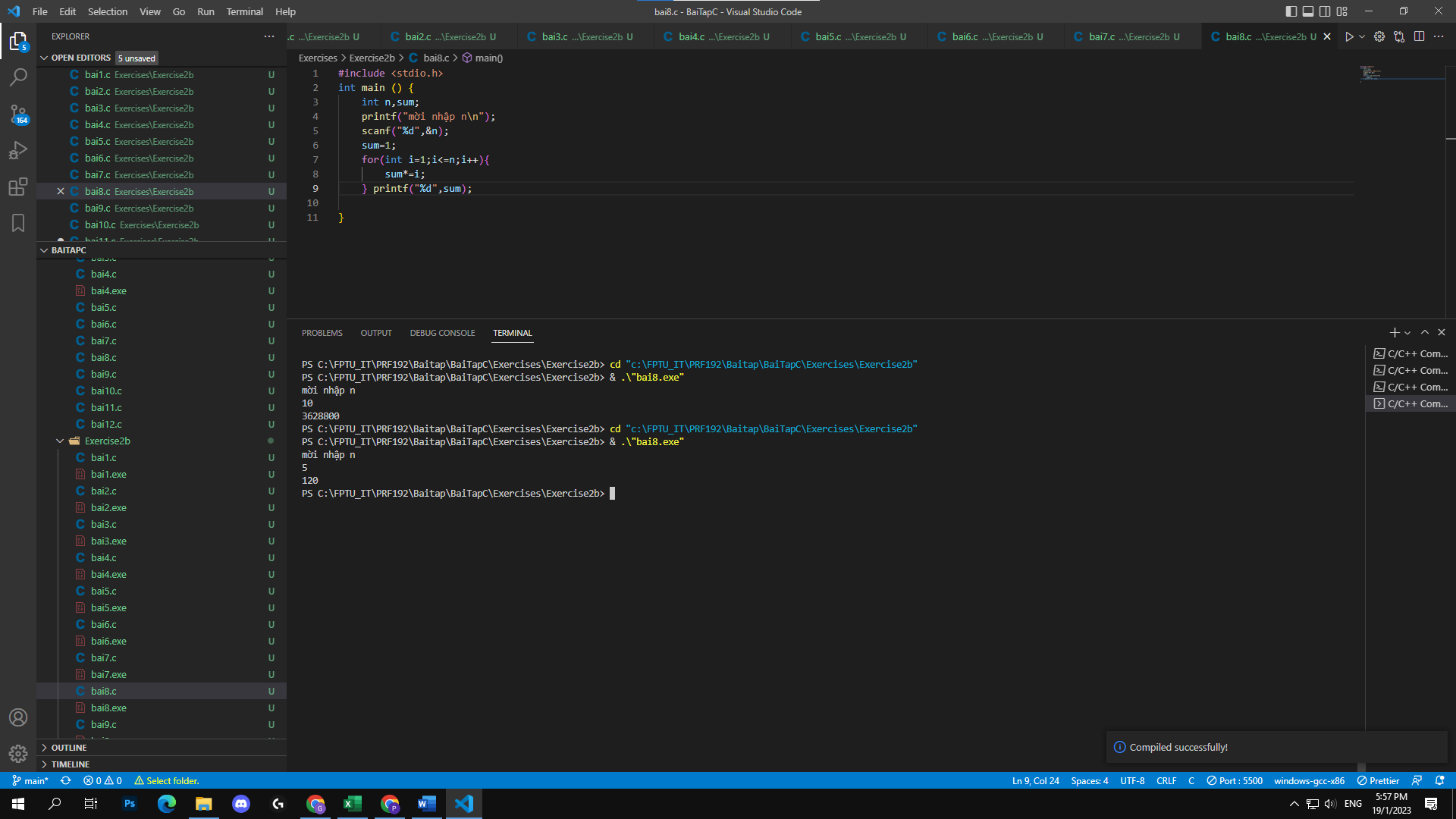
Write a program that uses a loop structure to calculate the sum of even numbers from *1* to *n*, where n is entered from the keyboard.

For example, input n = 10, the result is 30 (2 + 4 + 6 + 8 + 10 = 30).



**\* Exercise 5.8: Calculate product of digits of a number**

Write a program to find and display to screen the product of a natural number *n*, where n is entered from the keyboard**.**

****

**\* Exercise 5.9: Find the Armstrong numbers**

Armstrong number is a number that is equal to the sum of cubes of its digits. For example, 0, 1, 153, 370, 371 and 407 are the Armstrong numbers.

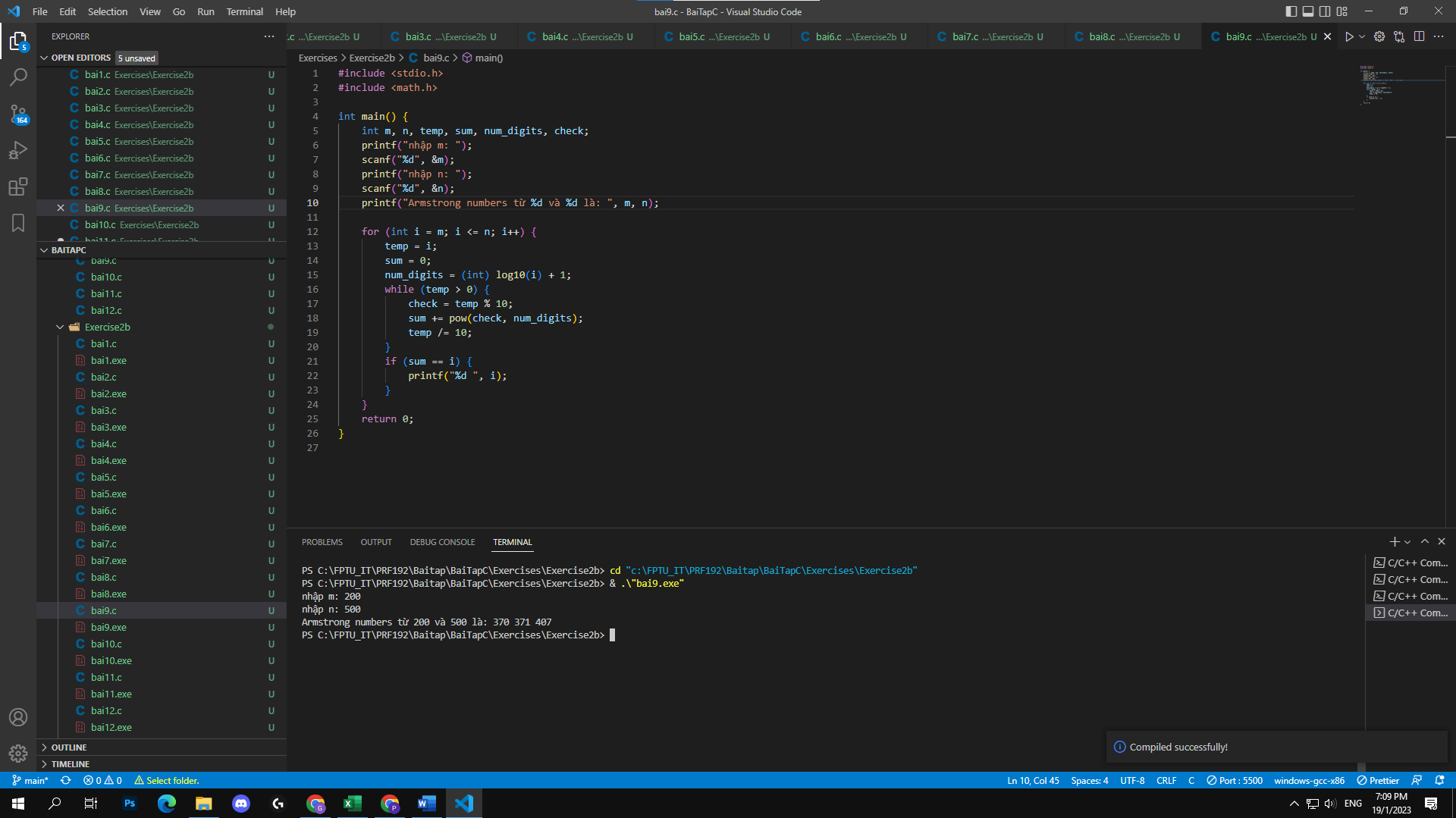
Write a program to input two natural numbers *m* and *n*, then find and display to screen all Armstrong numbers from *m* to *n*. The Armstrong numbers are separated by a space.

Example:

Enter number m: 200

Enter number n: 500

Armstrong numbers from 200 and 500 are: 307 371 407



**\* Exercise 5.10: Palindrome** **number**

A palindrome number is a number that reads the same forward or backward. For example, 121, 34543, 343, 131, 48984 are the palindrome numbers.

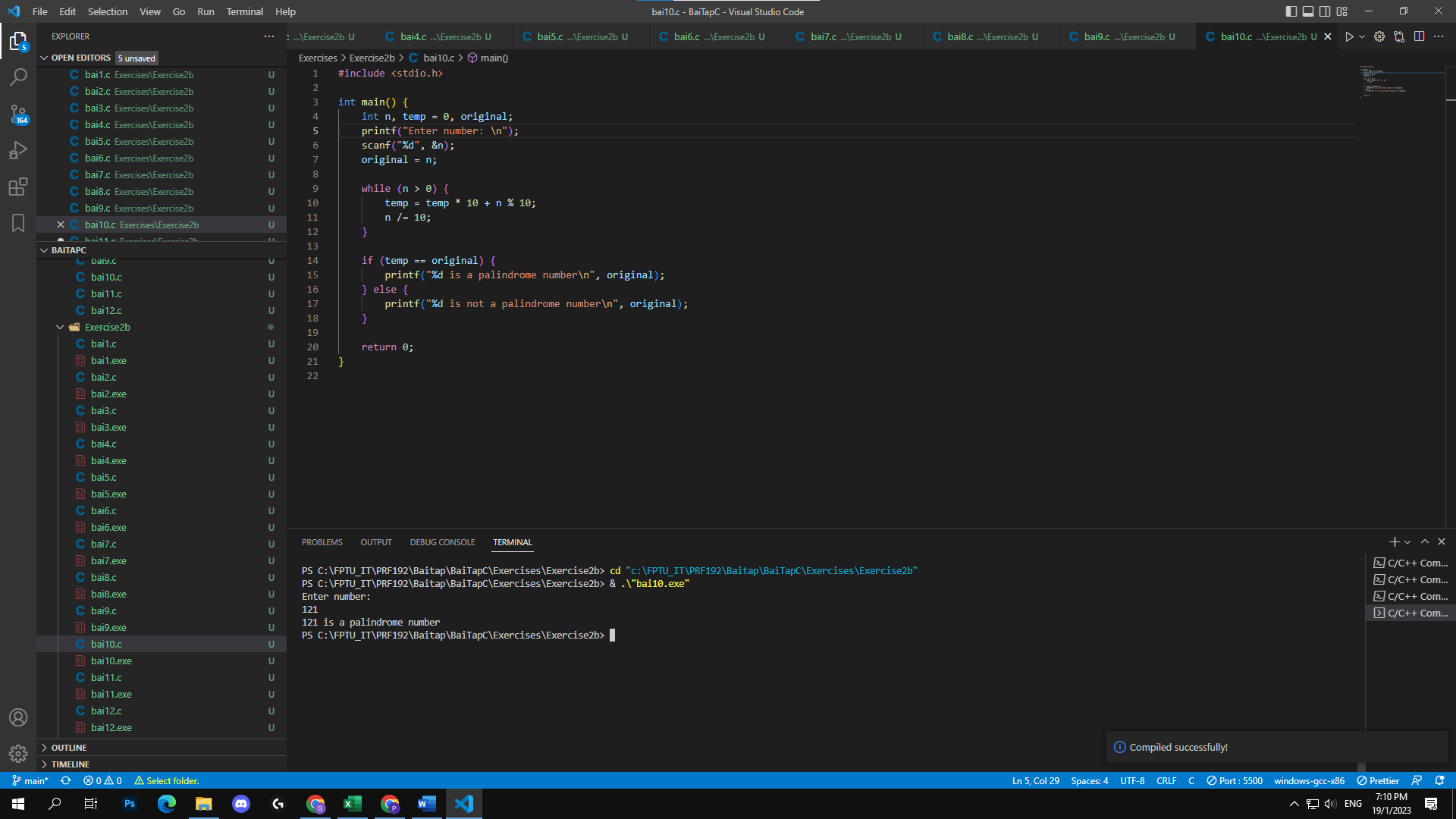
Write a program to check whether a number *n* is palindrome or not, where n is entered from the keyboard**.**

Example:

Enter number: 121

Result: 121 is a palindrome number

**Note:** To check if a number is a palindrome, firstly we need to reverse it and then compare the resulting number with the original number, if both are same then the number is palindrome, otherwise it is not.



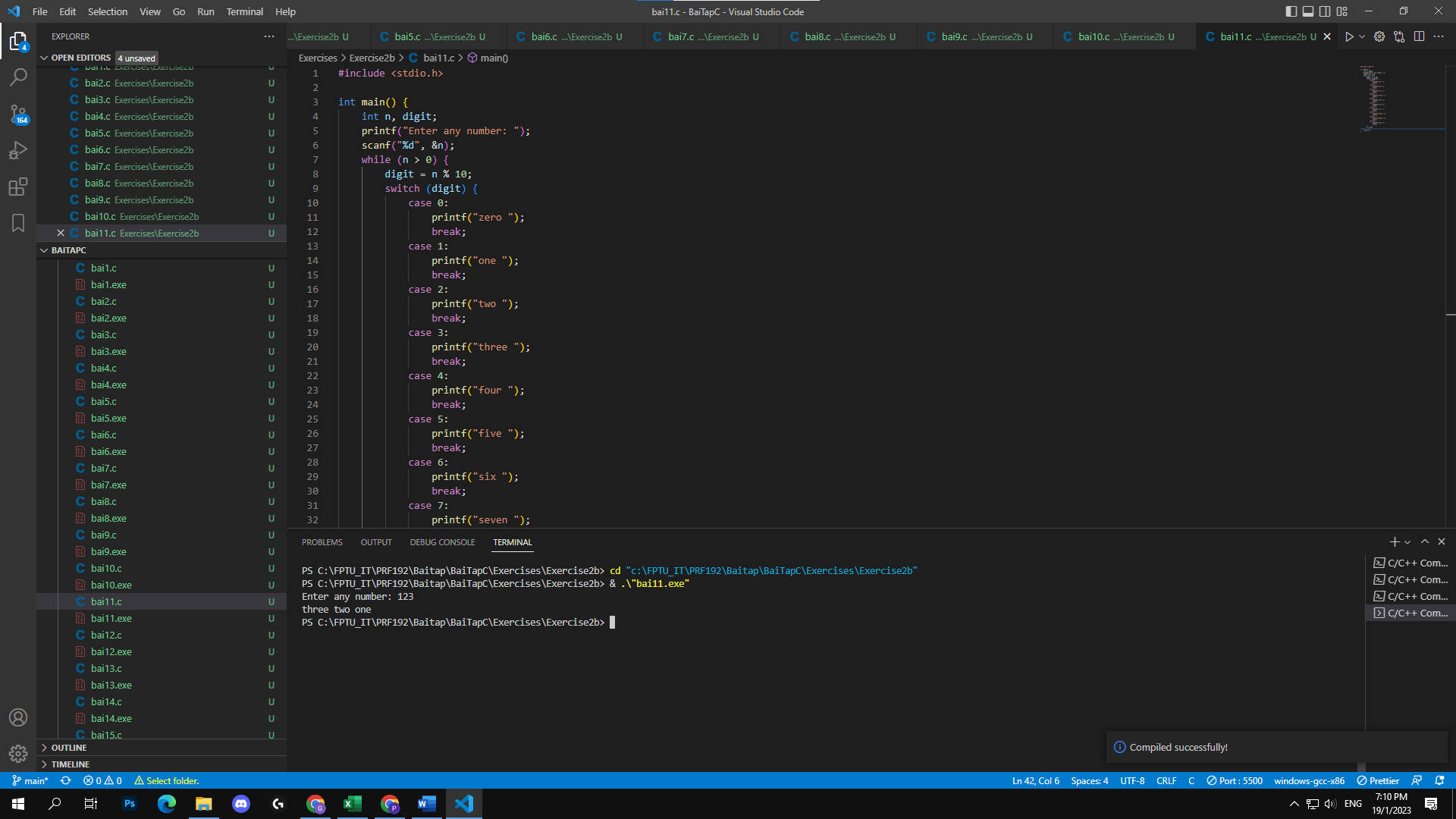
**\* Exercise 5.11: Print number in words**

Write a program that reads a natural number *n* from the keyboard and display all digits of the number *n* in English words.

Example:

Enter any number: 51324

five one three two four



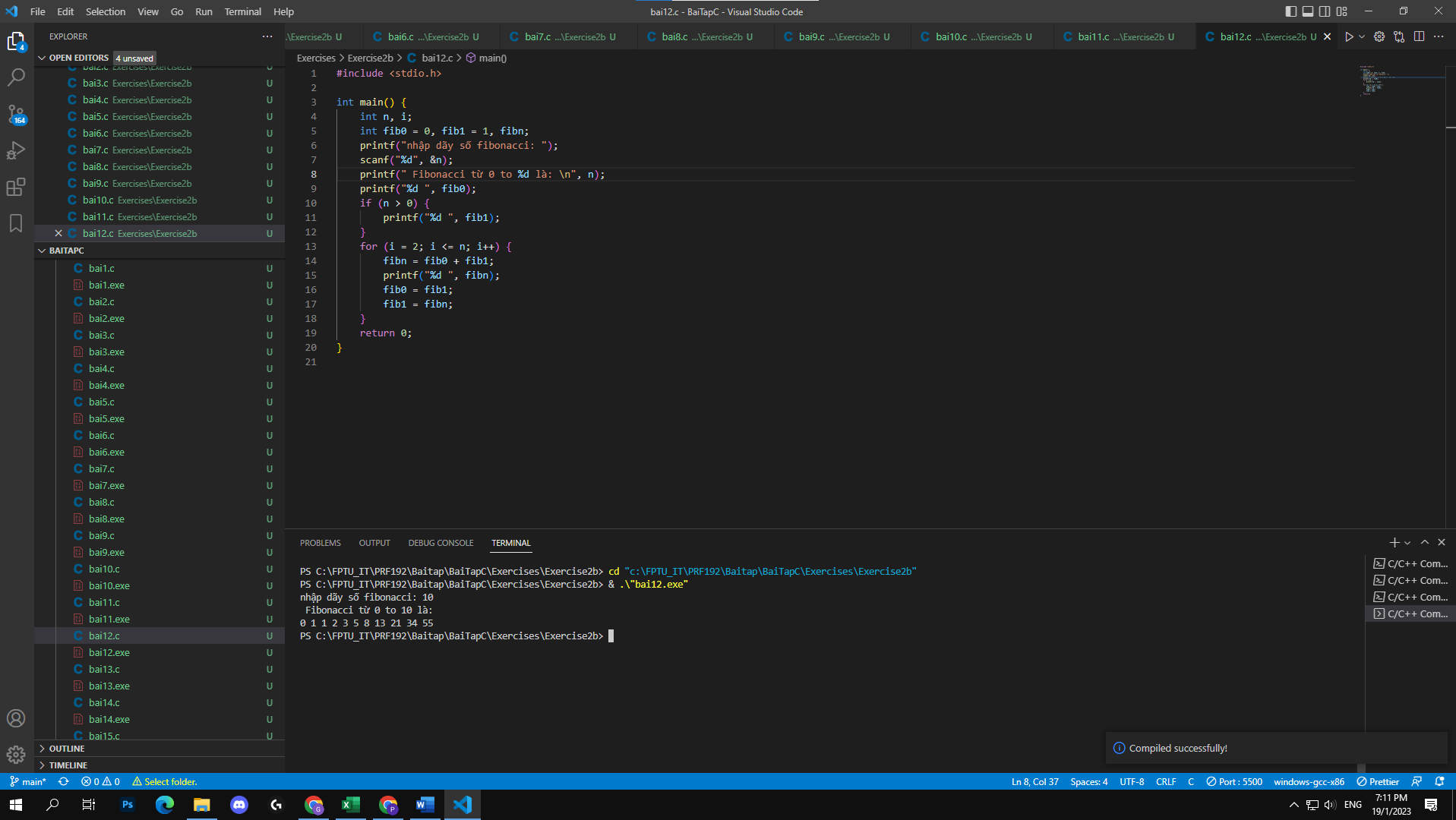
**\* Exercise 5.12: Fibonacci**

The Fibonacci sequence {Fn} is determined by the following formula:

F0=0, F1=1, Fn=Fn-1+Fn-2.

F0=0, F1=1, F2=1, F3=2, F4=3, F5=5, F6=8, F7=13, F8=21,…

Write a program that reads a number *n* from the keyboard and display all the Fibonacci numbers from 0 to n.



**\* Exercise 5.13: Matrix of signs "\*"**

Write a program to enter 2 positive integers *m* and *n* from the keyboard and then print to the screen a solid rectangle of size *m* × *n* with the **signs "\*"**.

Example:

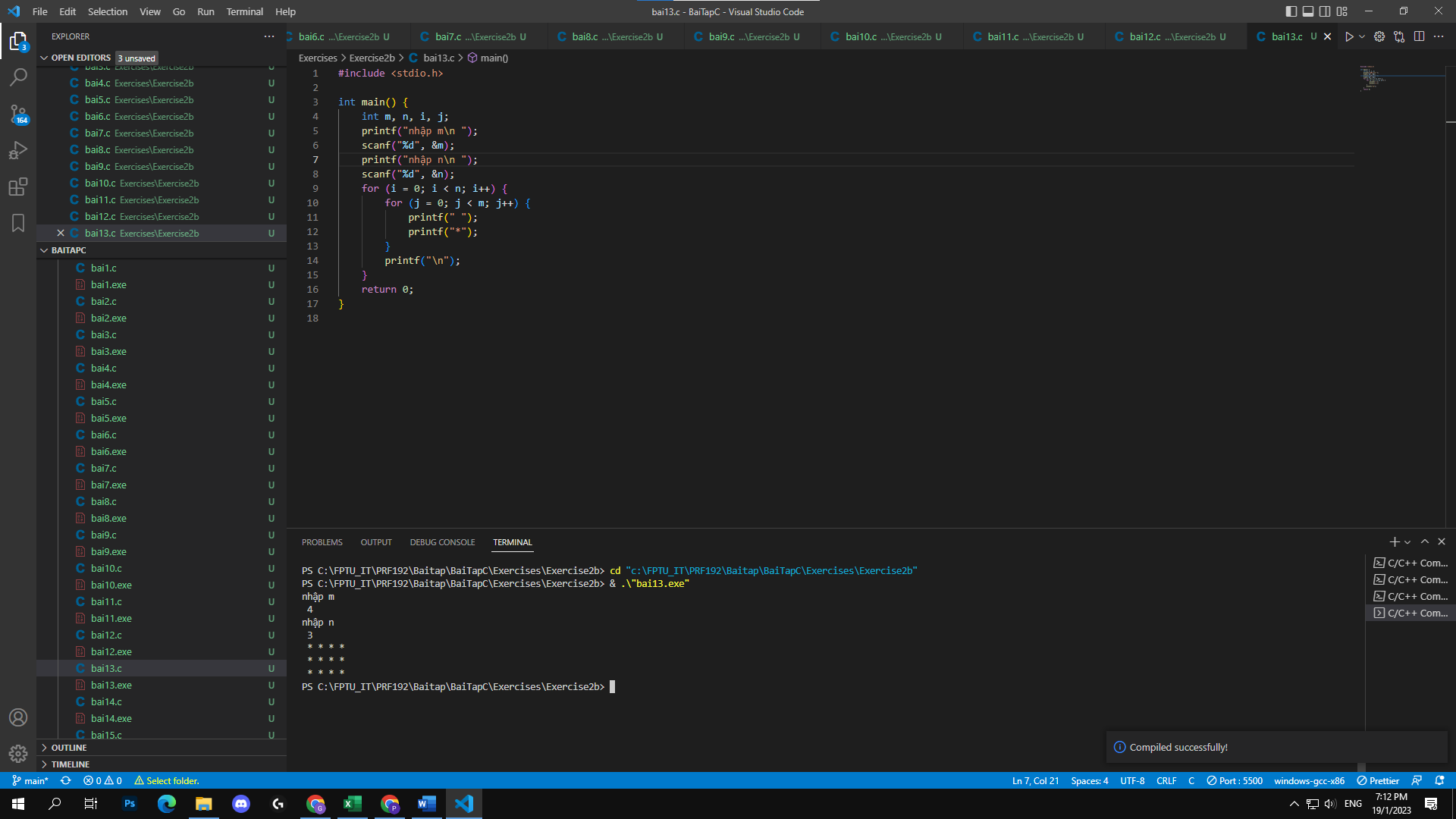
m=4, n=3

Result:

\* \* \* \*

\* \* \* \*

\* \* \* \*



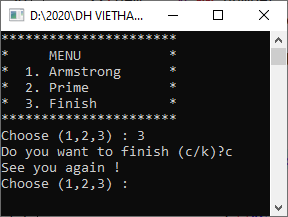
**\* Exercise 5.14: Building a program menu**

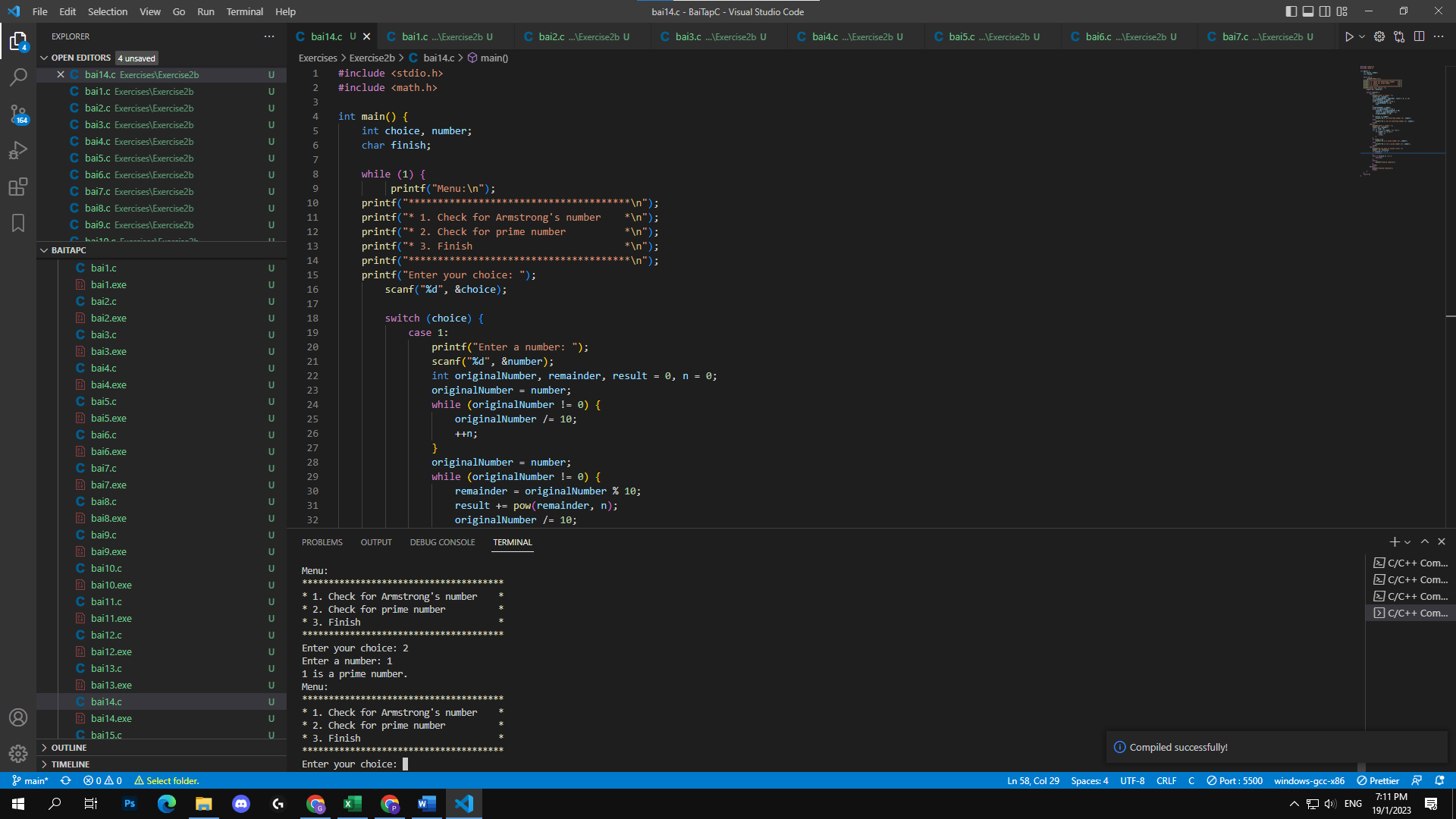
Write a program to build a menu as follows and perform the corresponding functions as shown below.

**If the user selects menu 1**: Ask to enter a number and check whether the number is a Armstrong's number or not.

**If the user selects menu 2**: Ask to enter a number and check whether the number is a prime number or not.

**If the user selects menu 3**: Display the question “Do you want to finish (c/k)?”. If the user enters “**c**”, exit the program; If the user enters “**k**”, allow to reselect the menu.





**\* Exercise 5.15: Finding the largest odd divisor**

Write a program that allows to:

- Enter a positive integer *n* (n>0) from the keyboard.

- Find the largest odd divisor of the positive integer n.

- If the largest odd divisor is found, the number will be displayed on the screen, otherwise, the character **“N**” will be displayed on the screen.

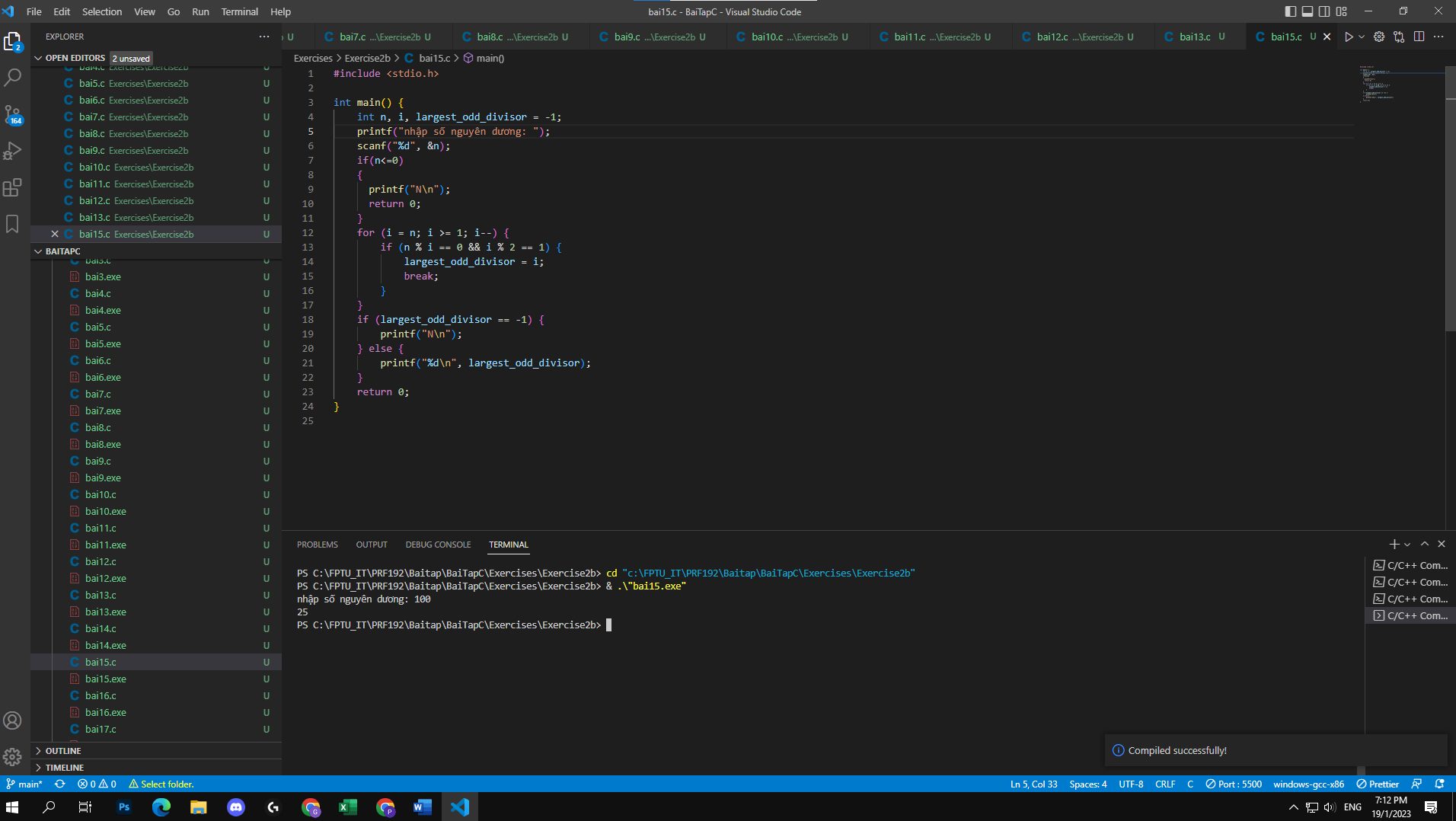
For example:

Input 1: 100

Output 1: 25

Input 2: 0

Output 2: N



**\* Exercise 5.16: Finding the odd digits**

Write a program that allows to:

- Enter a positive integer *n* from the keyboard.

- Find and display all odd digits of the number n.

- If there are no odd digits, display the character “**N**” to the screen.

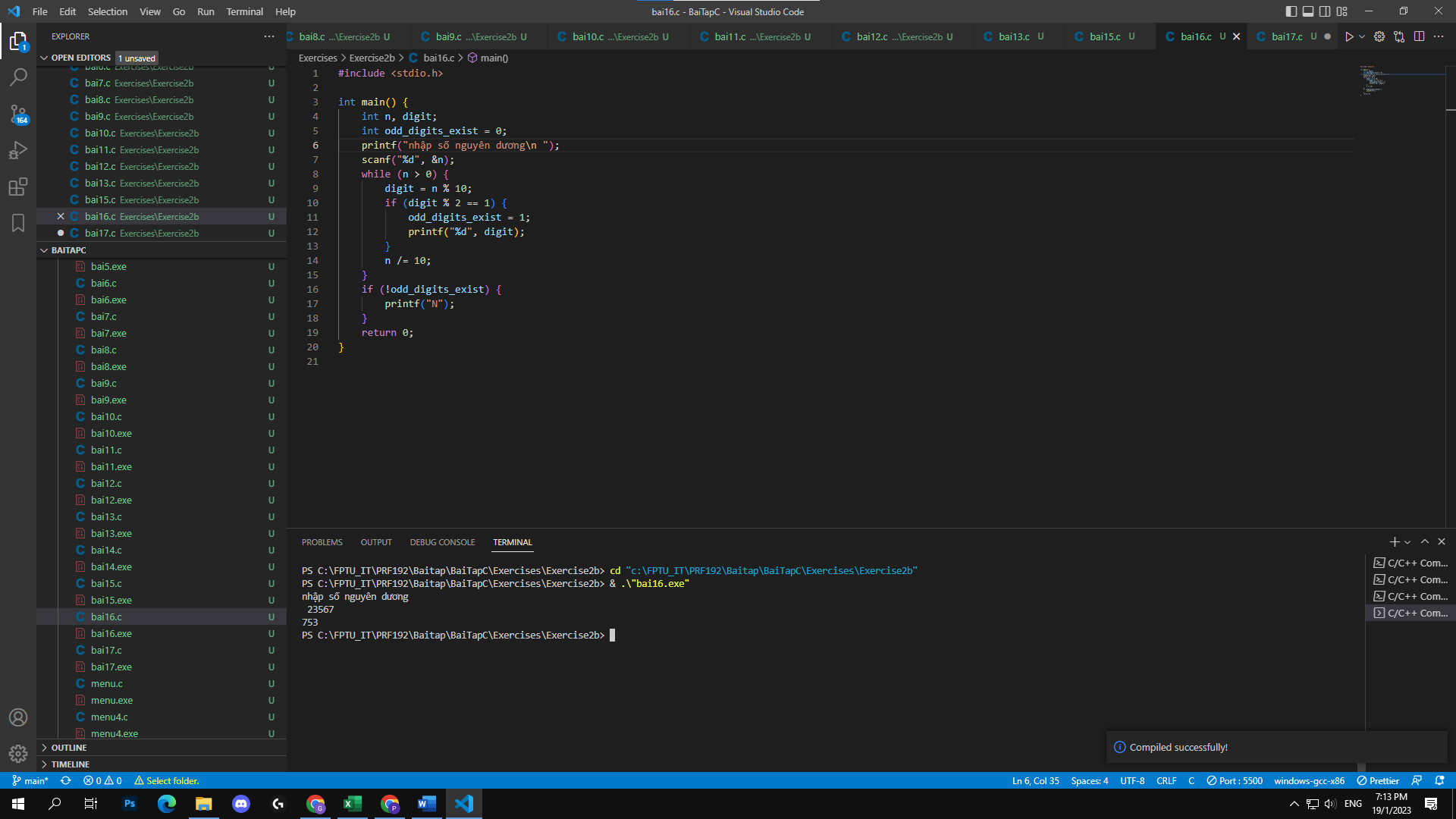
For example:

Input 1: 236574

Output: 357

Input 2: 2468

Output 2: N

****

**\* Exercise 5.17: Finding the largest k**

Write a program that allows to:

- Enter a positive integer *n* (n>0) from the keyboard.

- If *n* is not valid, display the character **N** to the screen and exit the program.

- If *n* is valid, find the largest positive integer k such that S(k) <= n, where S(k) is defined as follows: S(k) = 1 + 2 + 3 + … + k

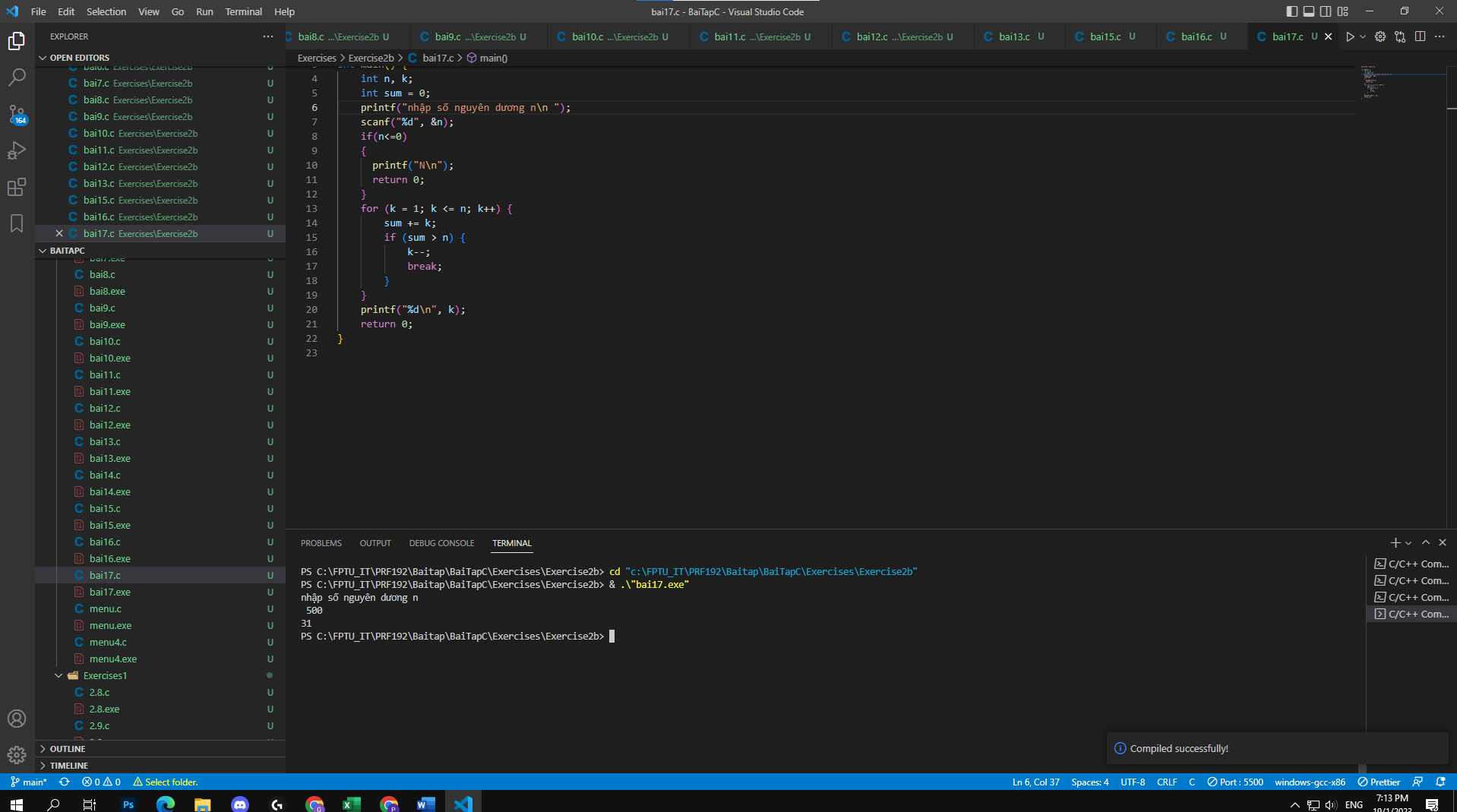
For example:

Input 1: 500

Output: 31

Input 2: -1

Output 2: N



**\* Exercise 5.18: File writing and reading**

Do the exercises from 5.4 to 5.8 using file manipulation functions.