**LAB 2 CSE325**

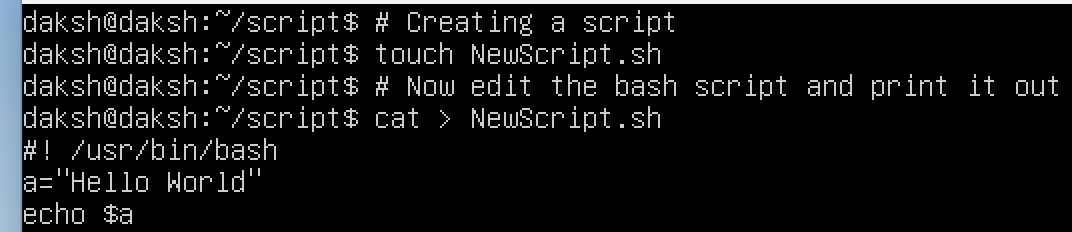
**Creating a bash script: -**

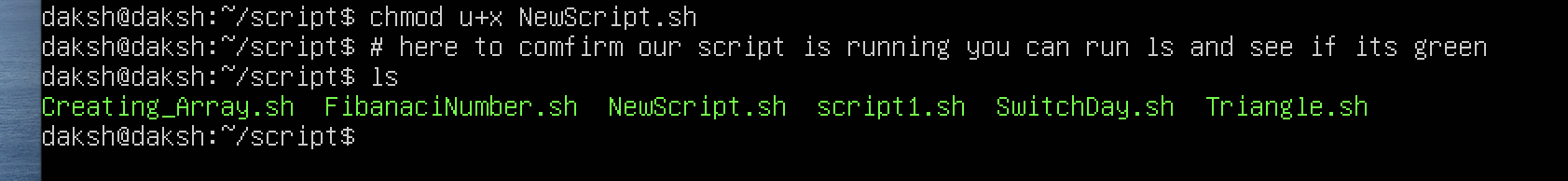
In this unit we will learn how to create a bash shell script. Basically we write a script for the work that we do again and again. So to save ourself from the hassle we create a script for that work which we have repeat again and again in our daily life. Script help us to complete these tasks easily.

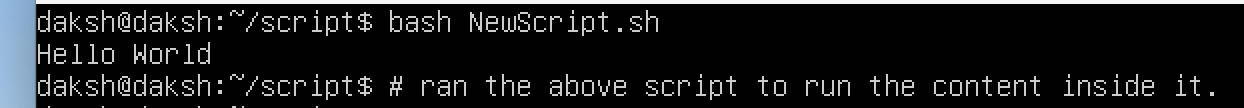
**Steps to create a shell script: -**

1. **We go to our terminal and create a file name like `Script\_name.sh`**
2. **Now your script file is ready and the extension `.sh` is for the script.**
3. **You can now open this script file with your editors here we are using the `vi` editor to edit all our bash scripts.**
4. **Once we are done with all this we have to open the bash script and write #! /usr/bin/bash/ here `#!` means shebang in bash and the path here is where your bash is located if your script is not working then this means your bash is on some other path to check the path we can simply use the command `which bash` and that will return the path of your bash.**
5. **After this open your file and start editing it once you are done then just save it.**
6. **After the creation and the editing of file you have to give your `user` the permission to edit and run the bash script which we will give by running the command `chmod u+x`.**
7. **Once you are done with all the above steps you can run your bash script by running the command `bash File\_name.sh`.**

**Screenshots: -**

****

****

****

Note: - If you have the root access then there’s no need for the chmod command to run the bash scripts

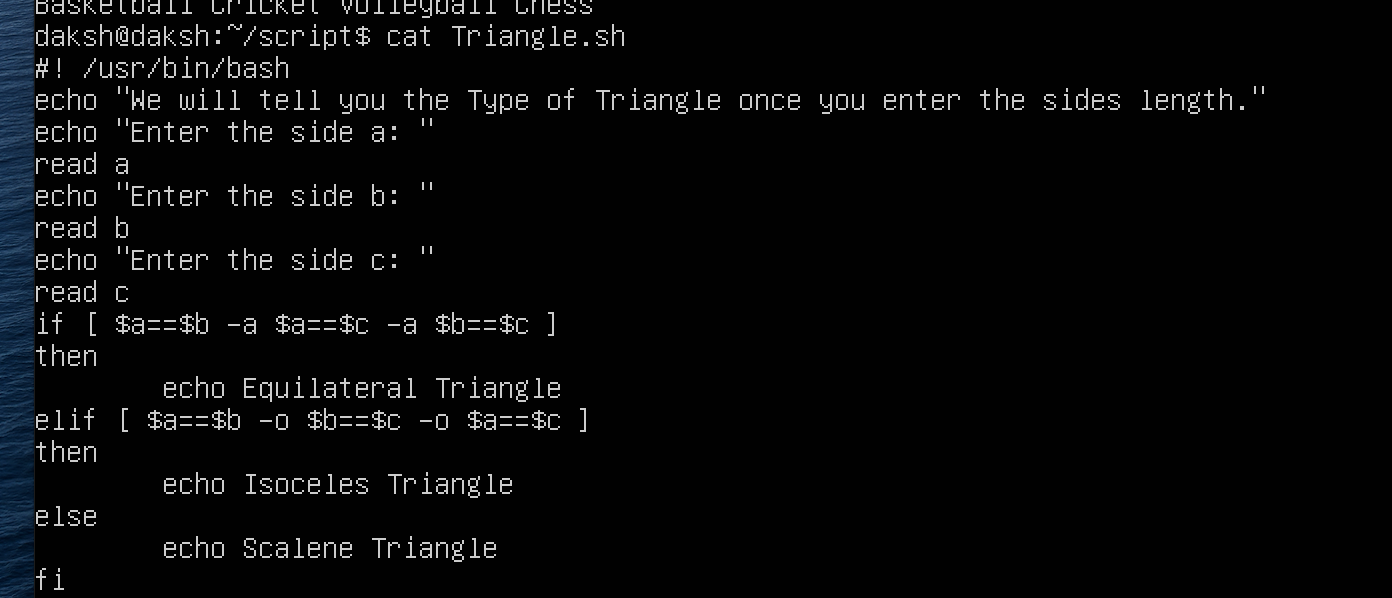
**Arithmetic Operators:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Use** | **Example** |
| **+** | **Addition** | **It adds two operands** | **Results= a+b** |
| **-** | **Subtraction** | **It subtracts second operand from the first one** | **Result= a-b** |
| **\*** | **Multiplication** | **Multiply two operands** | **Result= a\*b** |
| **/** | **Division** | **Return the quotient after diving the number** | **Result= a/b** |
| **%** | **Modulo** | **Returns the remainder of the number once it is divided** | **Result= a%b** |
| **+=** | **Increment by constant** | **Returns the result with the addition of the constant that is given** | **Result=**  **x=13**  **x+=3; Now x=16** |
| **-=** | **Decrement by constant** | **Returns the result with the subtraction by the given constant** | **Result=**  **x=13**  **x-=3; Now x=10** |
| **/=** | **Divide by constant** | **Returns the result by diving the number with the given constant** | **Result=**  **x=15**  **x/=5; Now x=3** |
| **%=** | **Remainder by diving the constant** | **Return the result with given constant by giving the remainder by diving the num with constant** | **Result=**  **x=31**  **x%=3; Now x=1** |
| **\*\*** | **Exponential** | **Return the number by the power of constant** | **Result=**  **3\*\*2=9** |
|  |  |  |  |

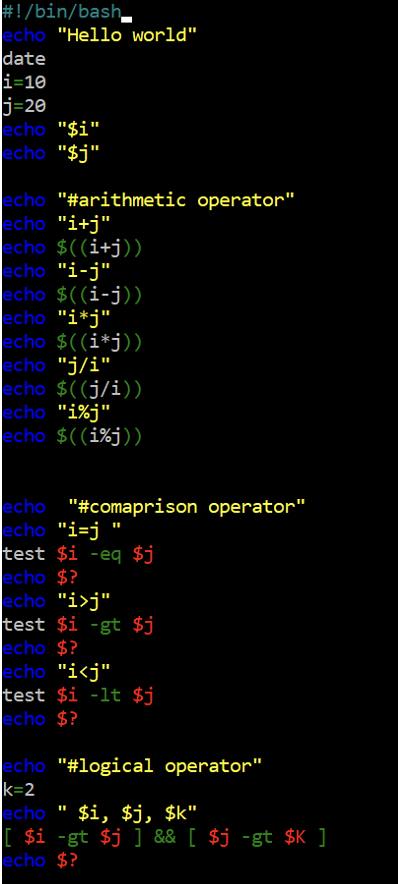
**Comparison Operators: -**

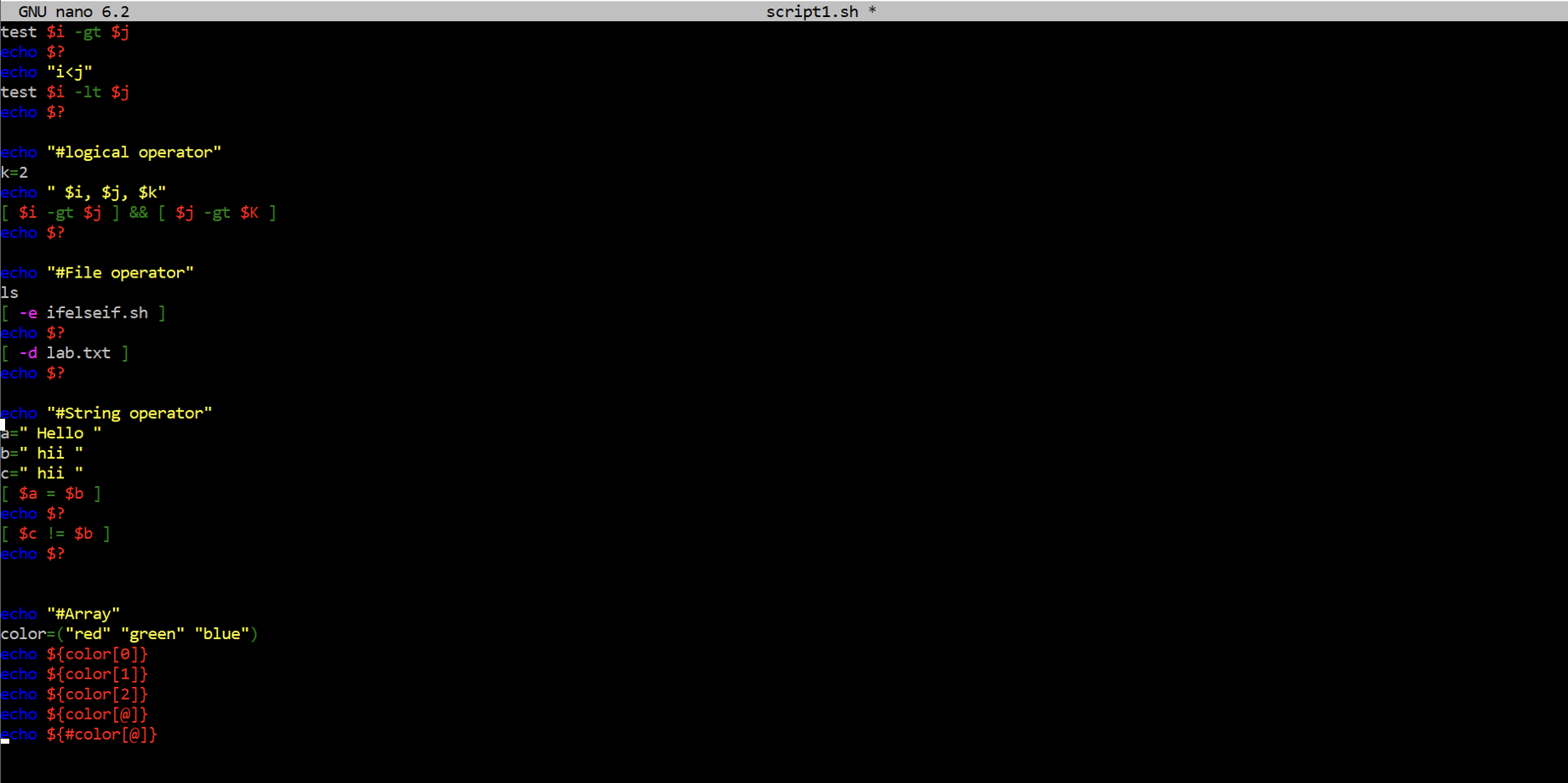
|  |  |  |
| --- | --- | --- |
| **OPERATIONS** | **SYNTAX** | **EXPLANATION** |
| Equality | Num1 -eq Num2 | Is num1 equal to num2 |
| Less than equal to | Num1 -le Num2 | Is num1 less than equal to num2 |
| Less than | Num1 -lt Num2 | Is num1 less than num2 |
| Greater than equal to | Num1 -ge Num2 | Is num1 greater than equal to num2 |
| Greater than | Num1 -gt Num2 | Is num1 greater than num2 |
| Not equal to | Num1 -ne Num2 | Is num1 not equal to num2 |

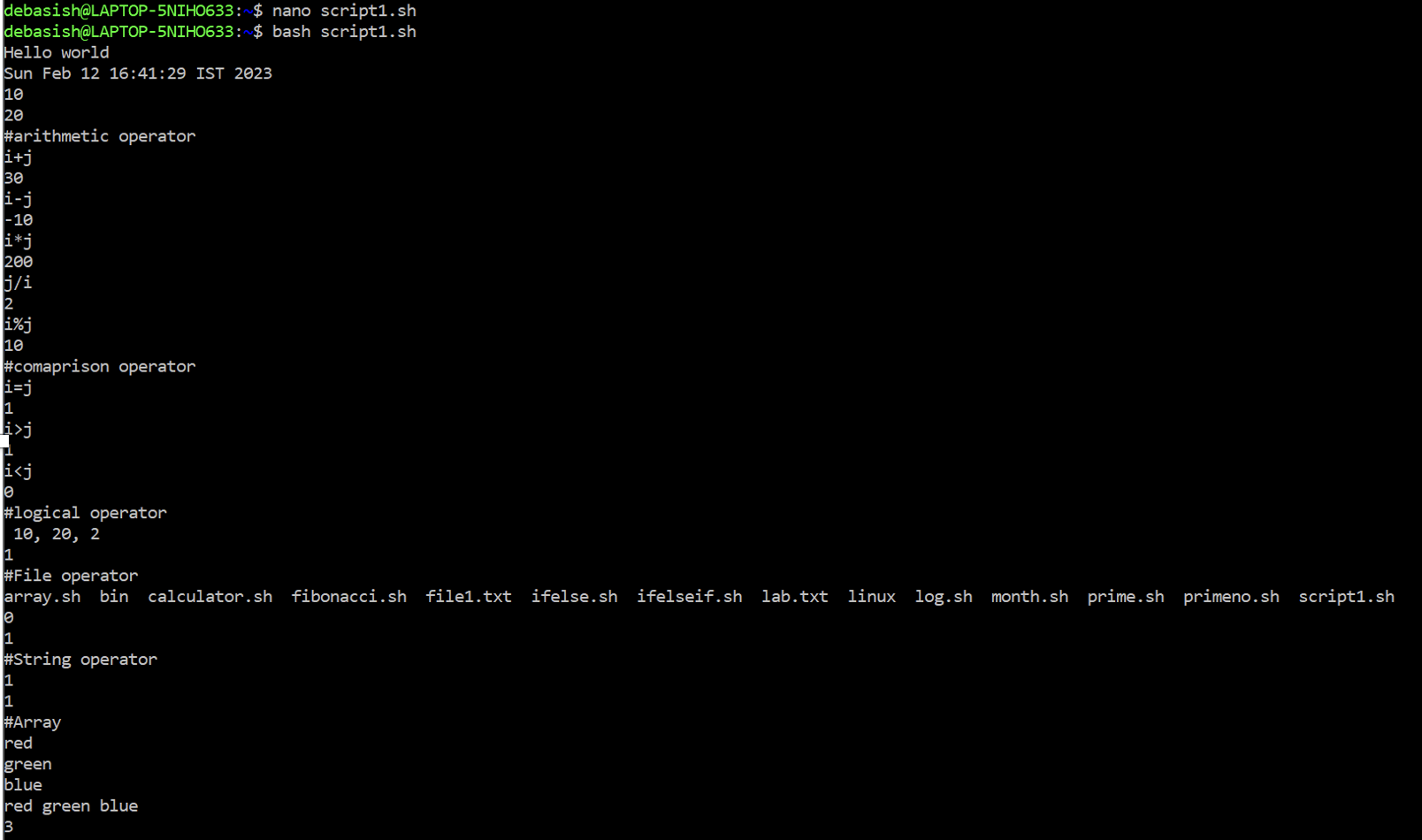
**Using all the comparison operators**

****

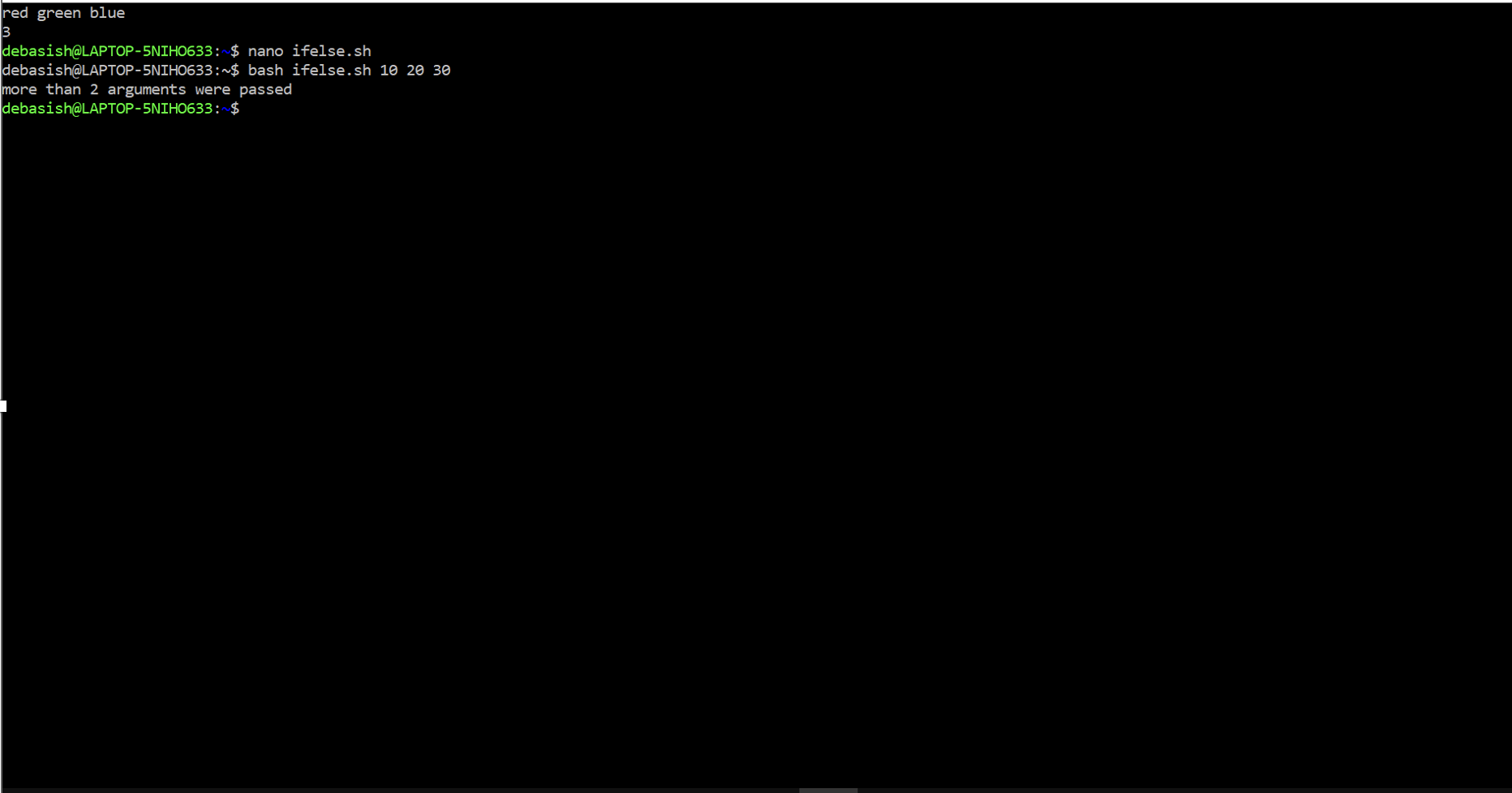
Screenshots of all processes :-

****

****

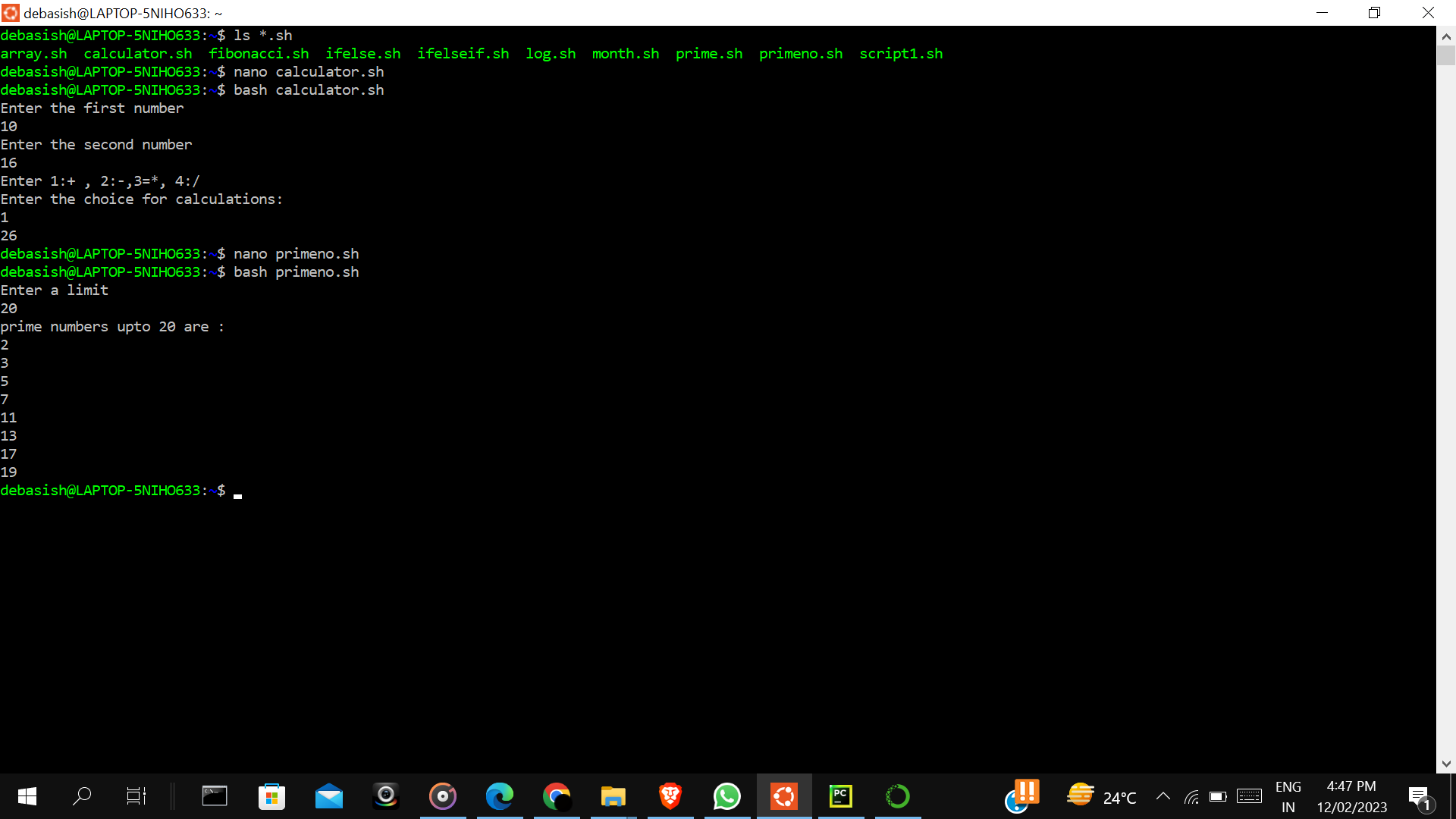
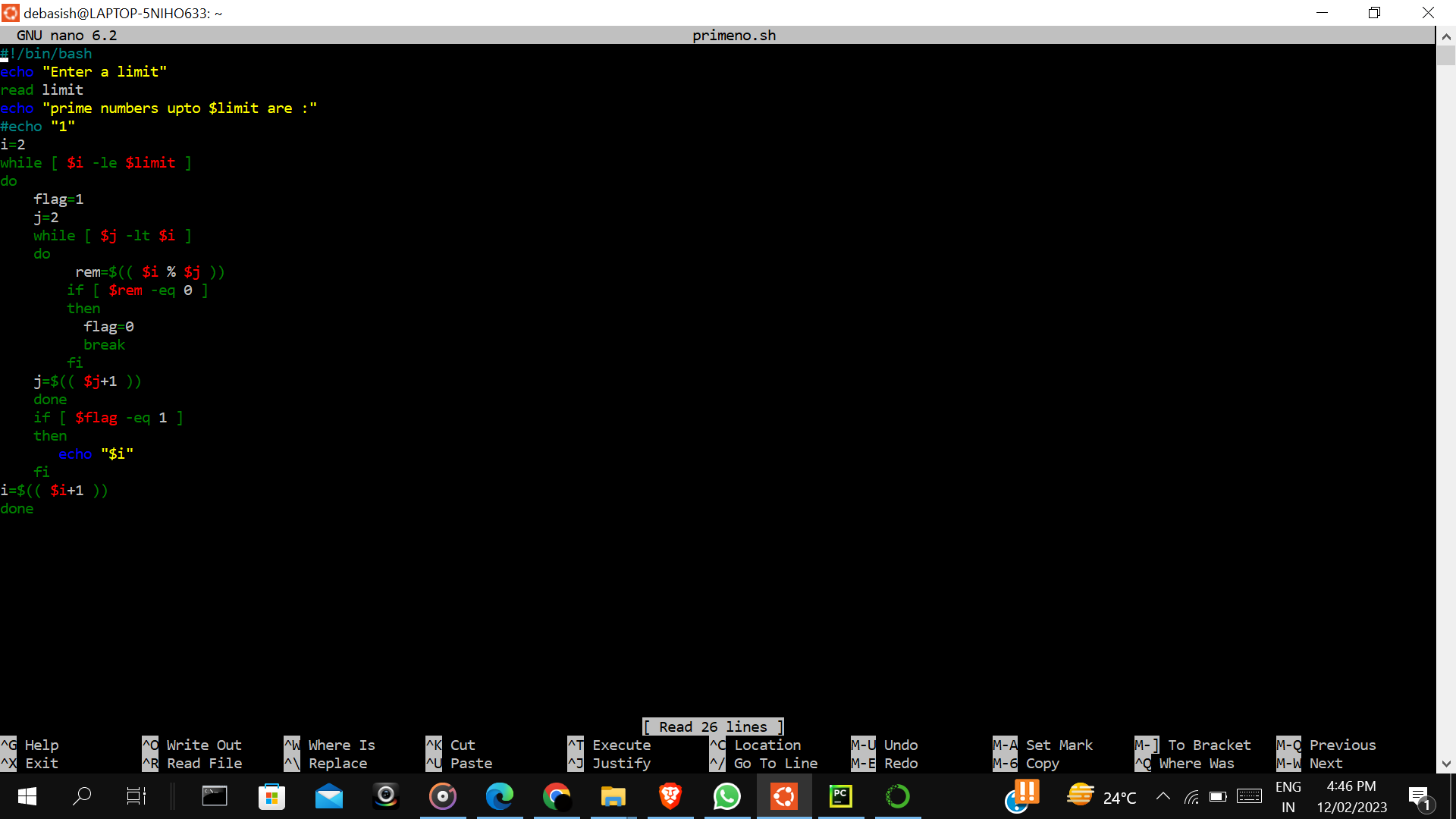
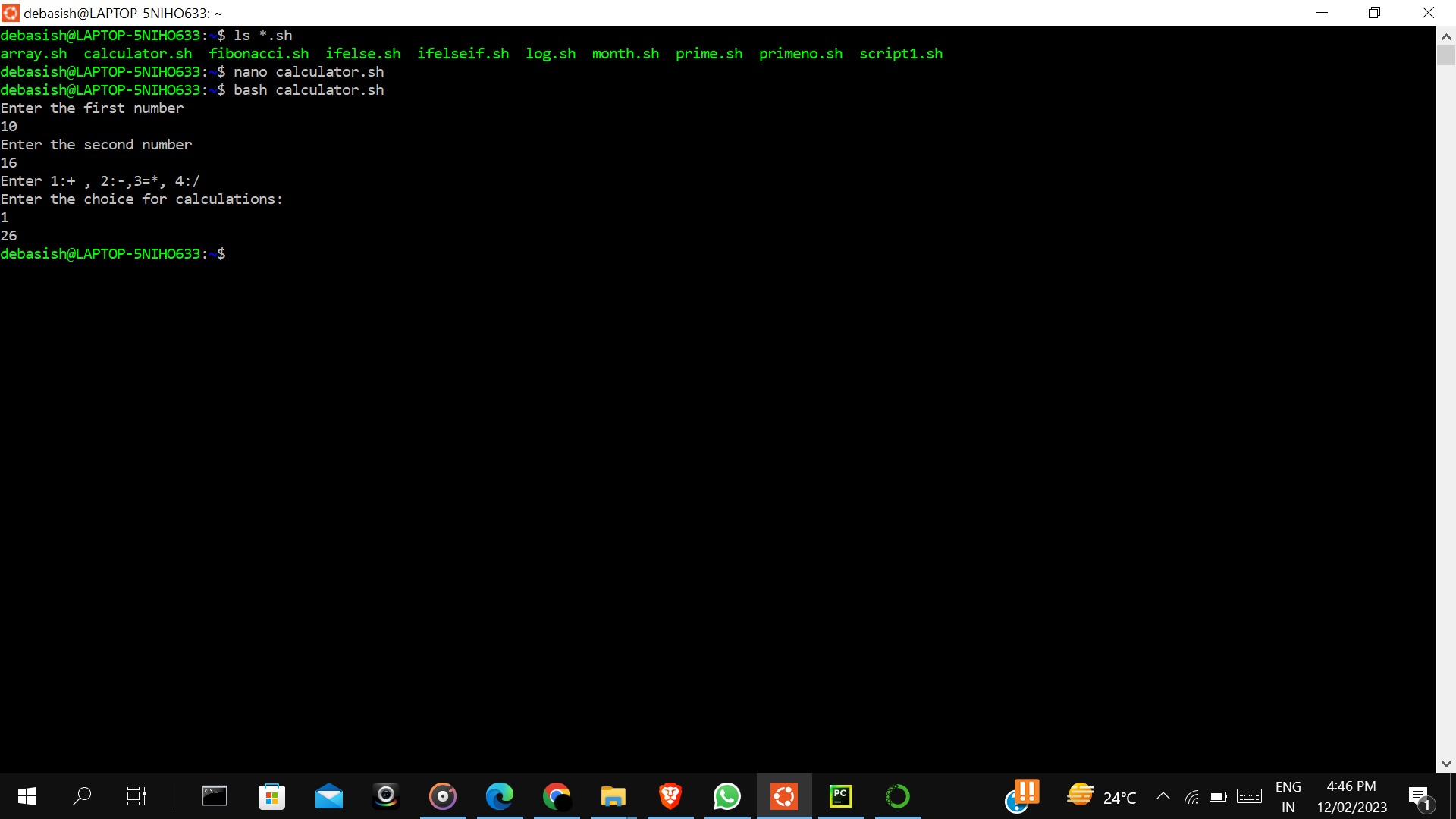
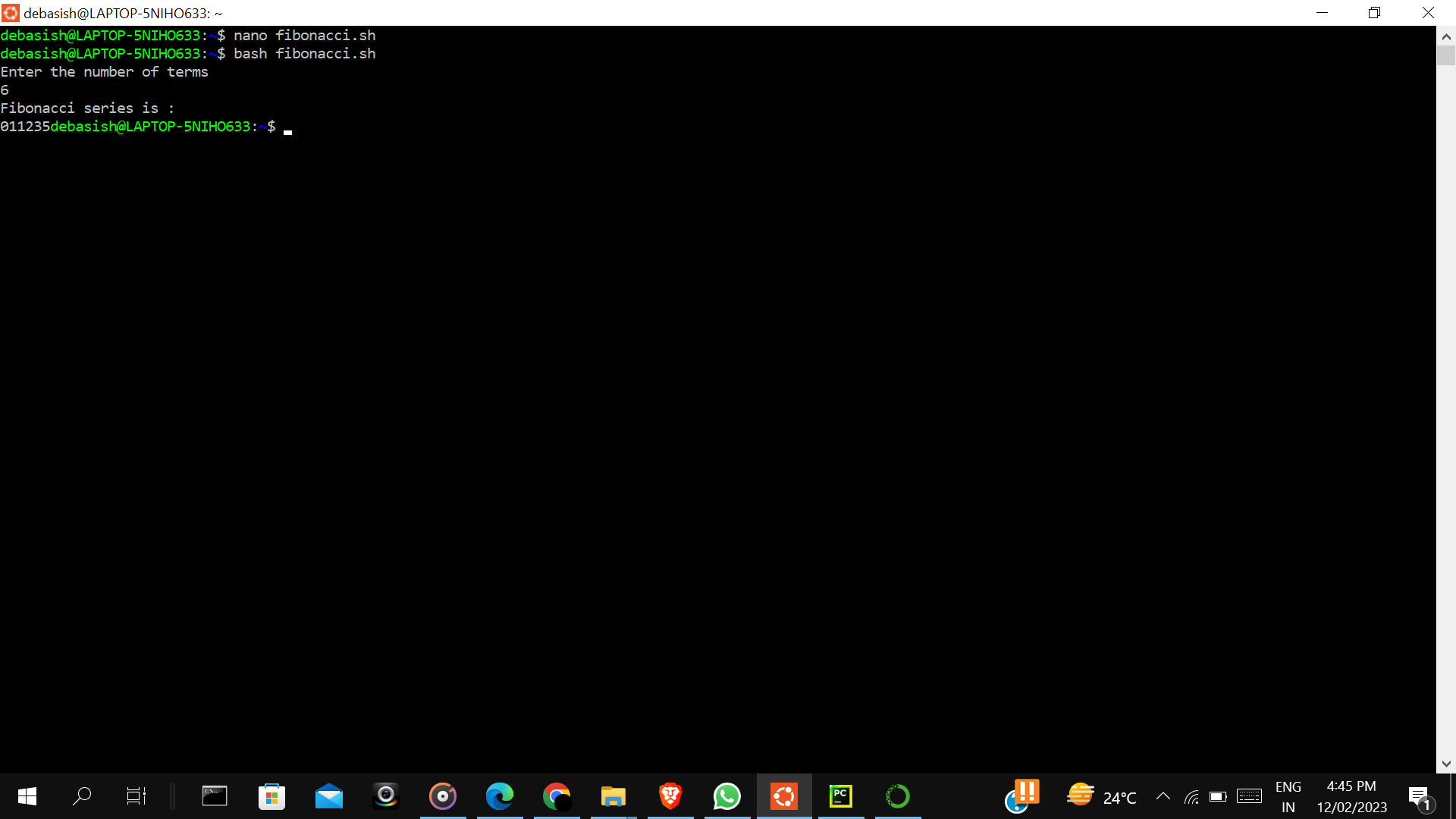
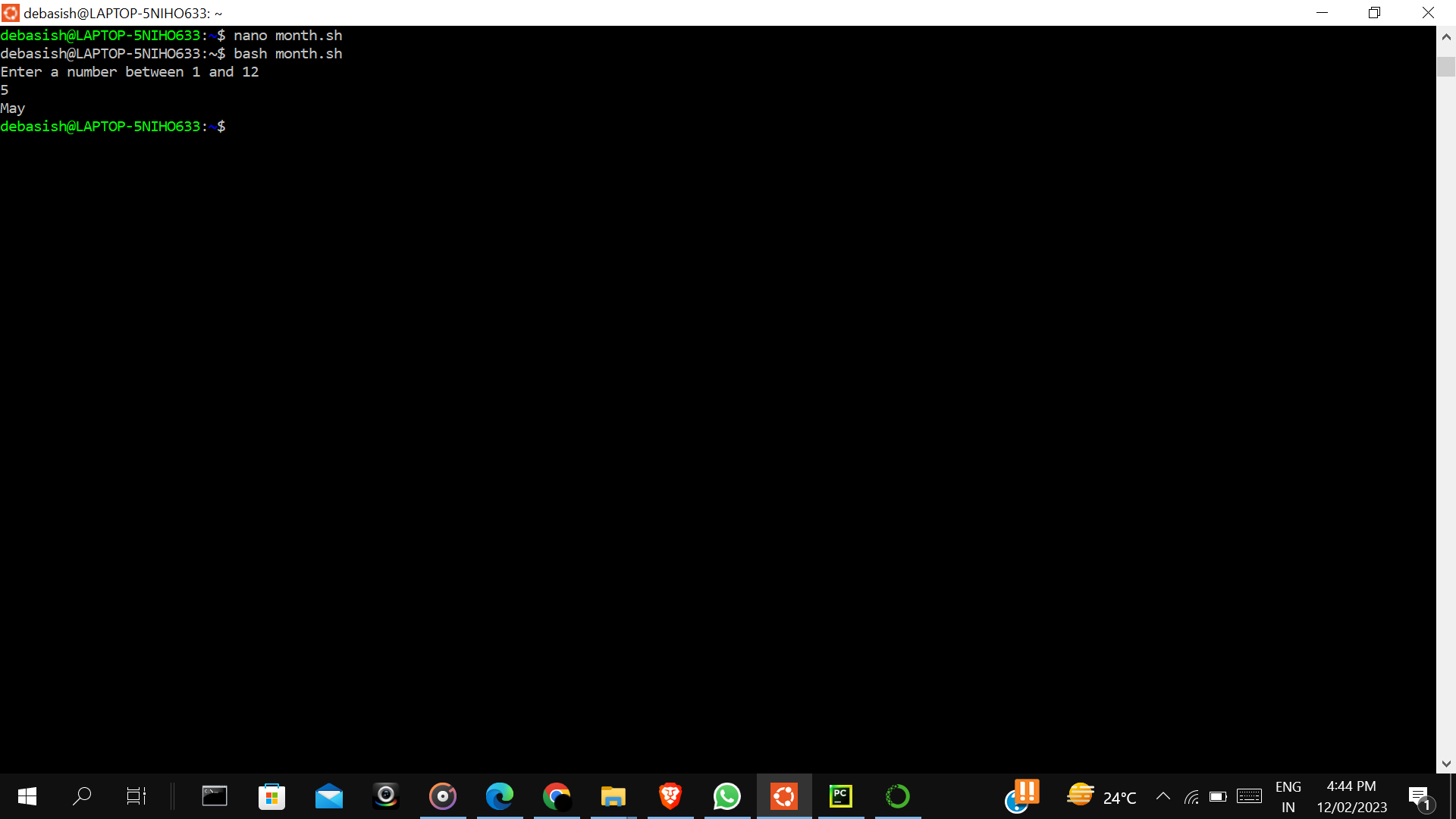
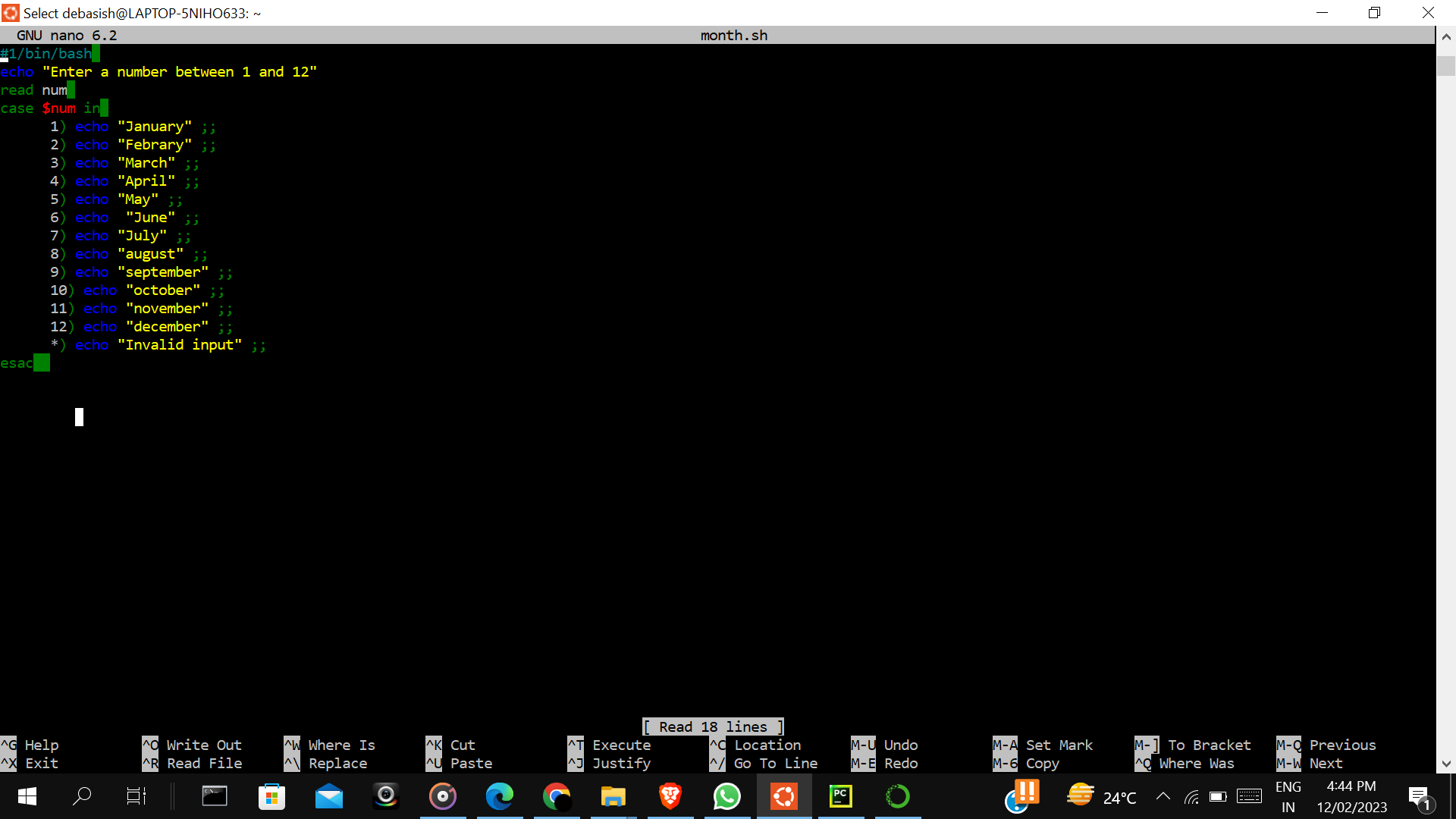
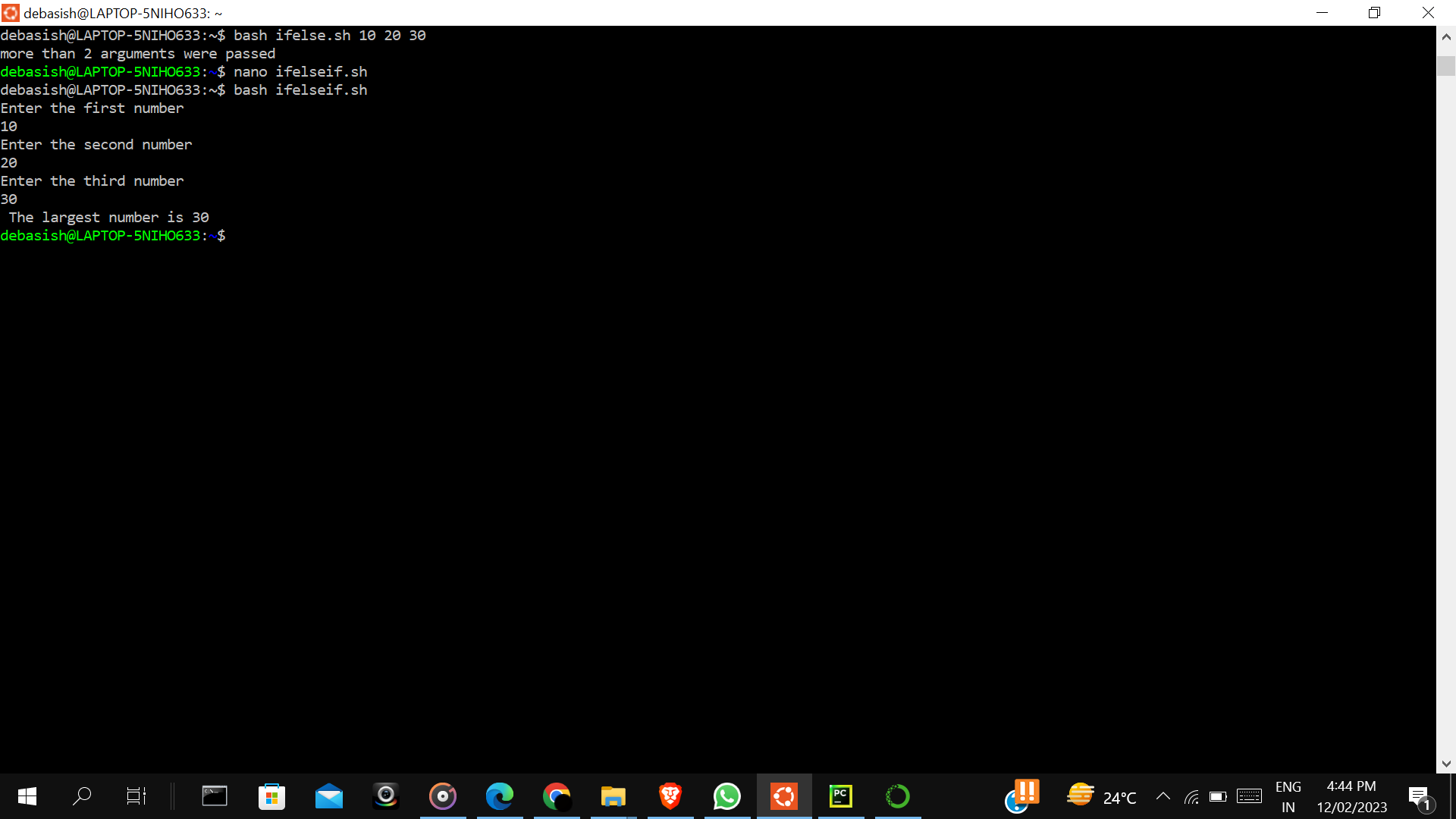
****

****

****

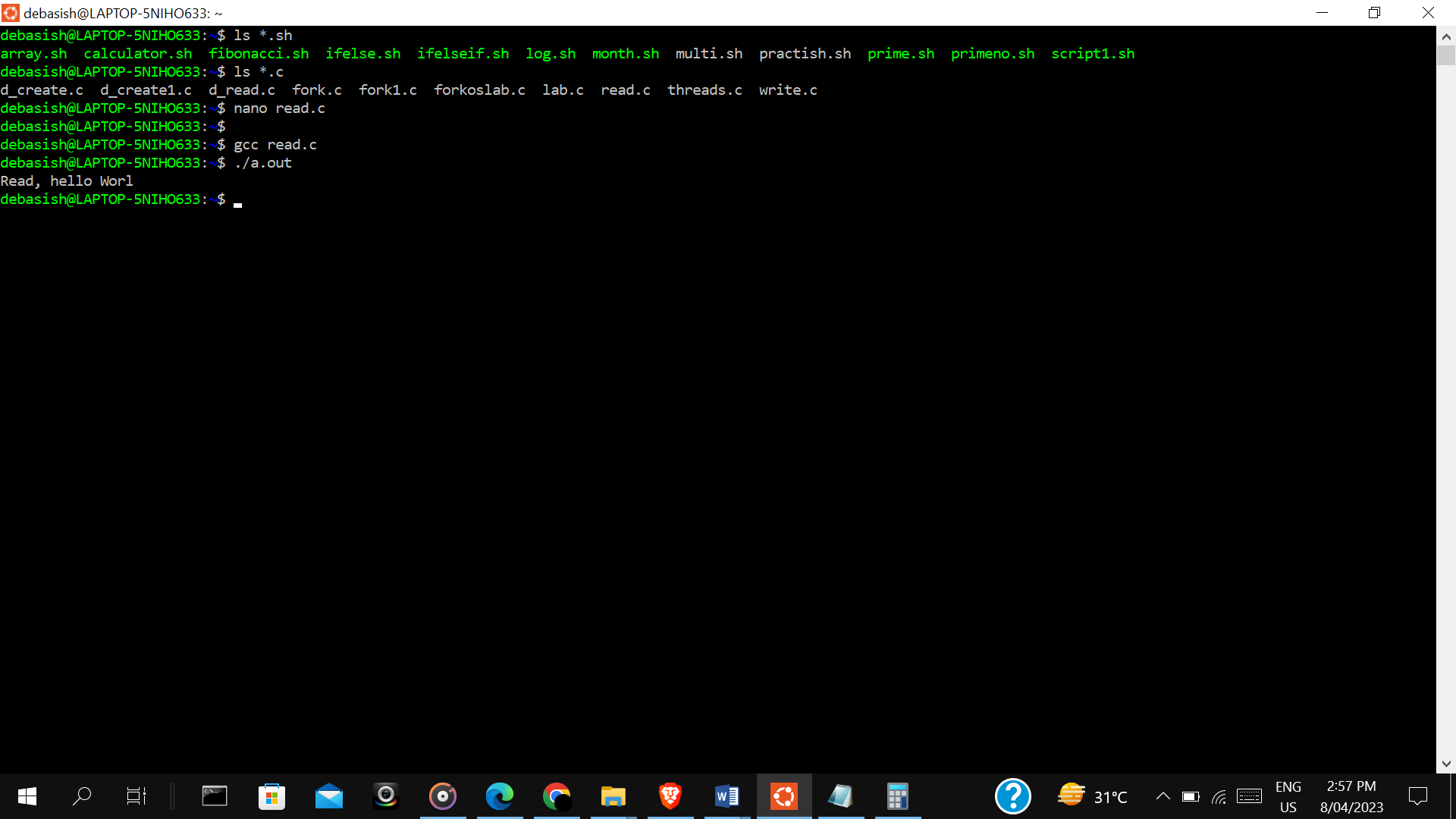
**A picture containing graphical user interface

Description automatically generated**

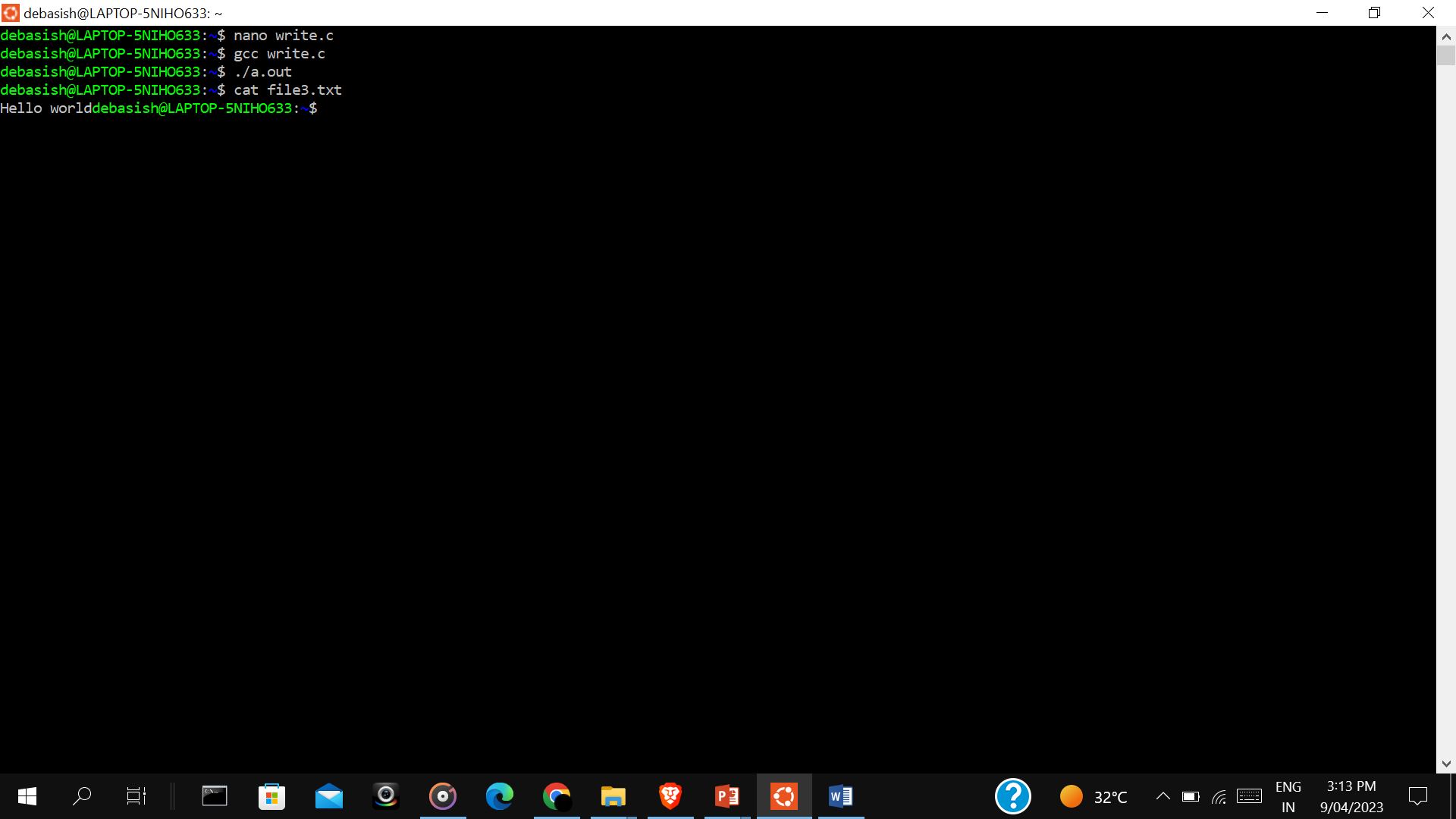
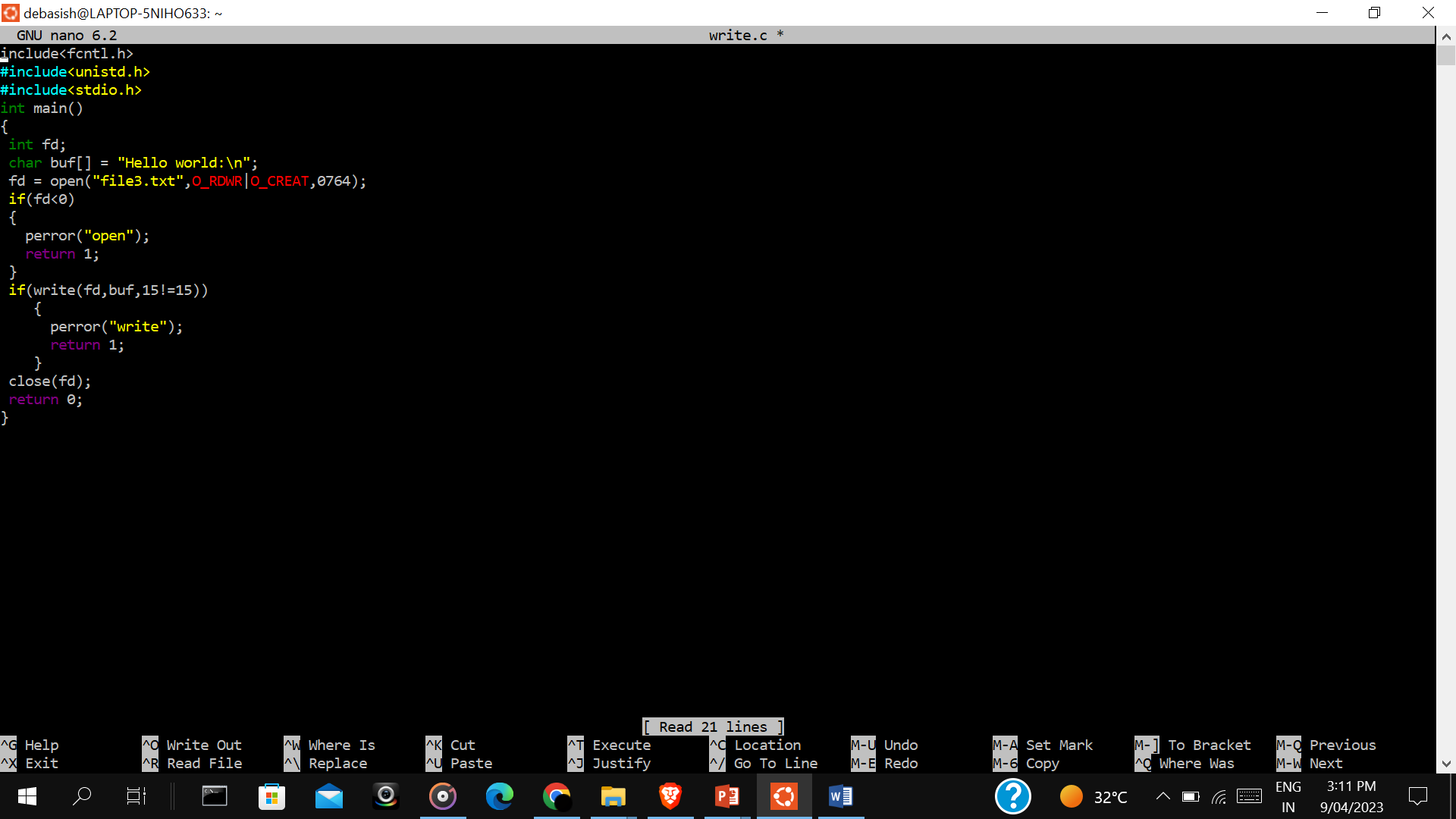


**Experiment 3: System call**

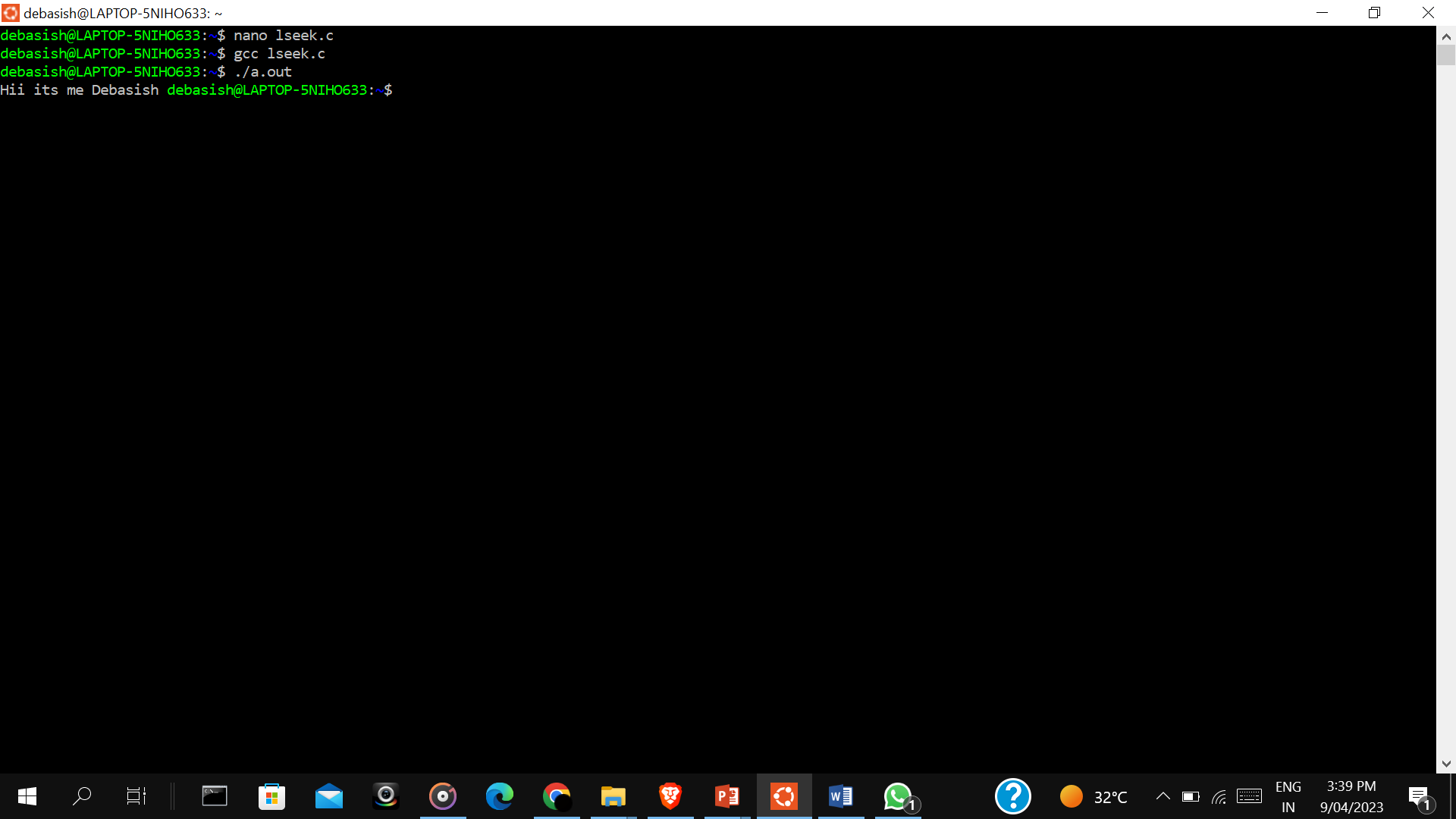
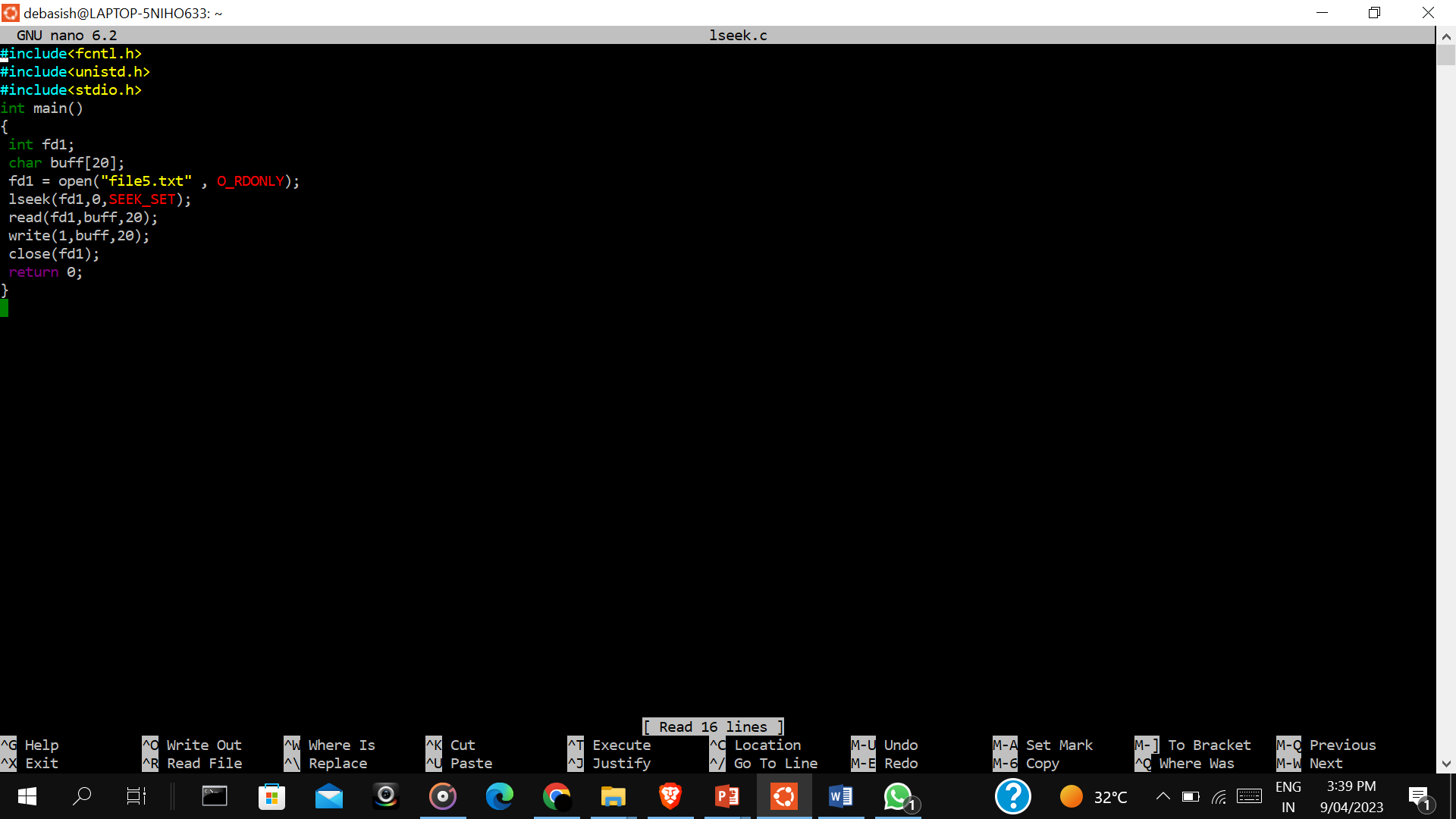
**Read.c:-**



Write.c:-

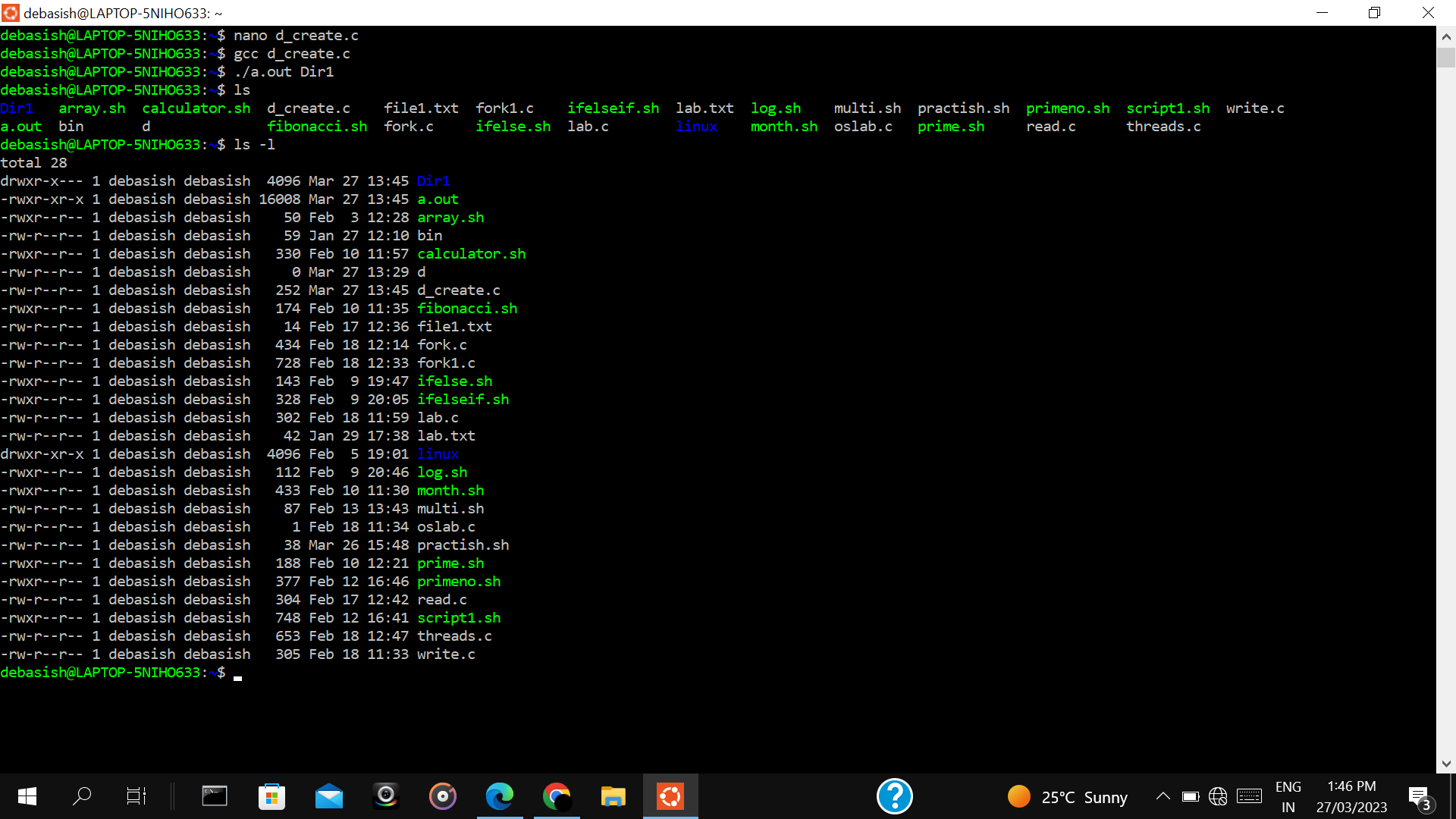
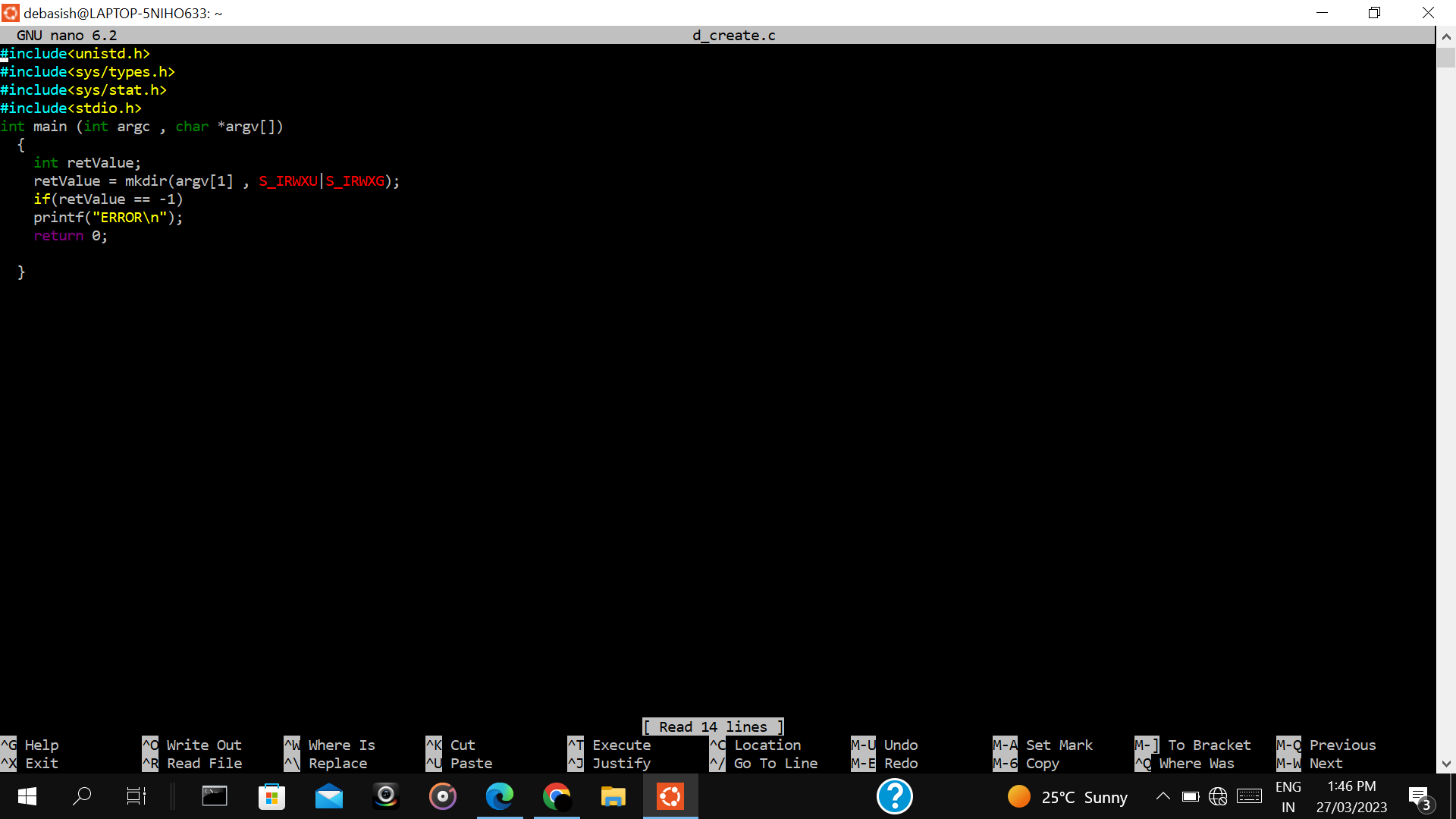


Lseek.c:-

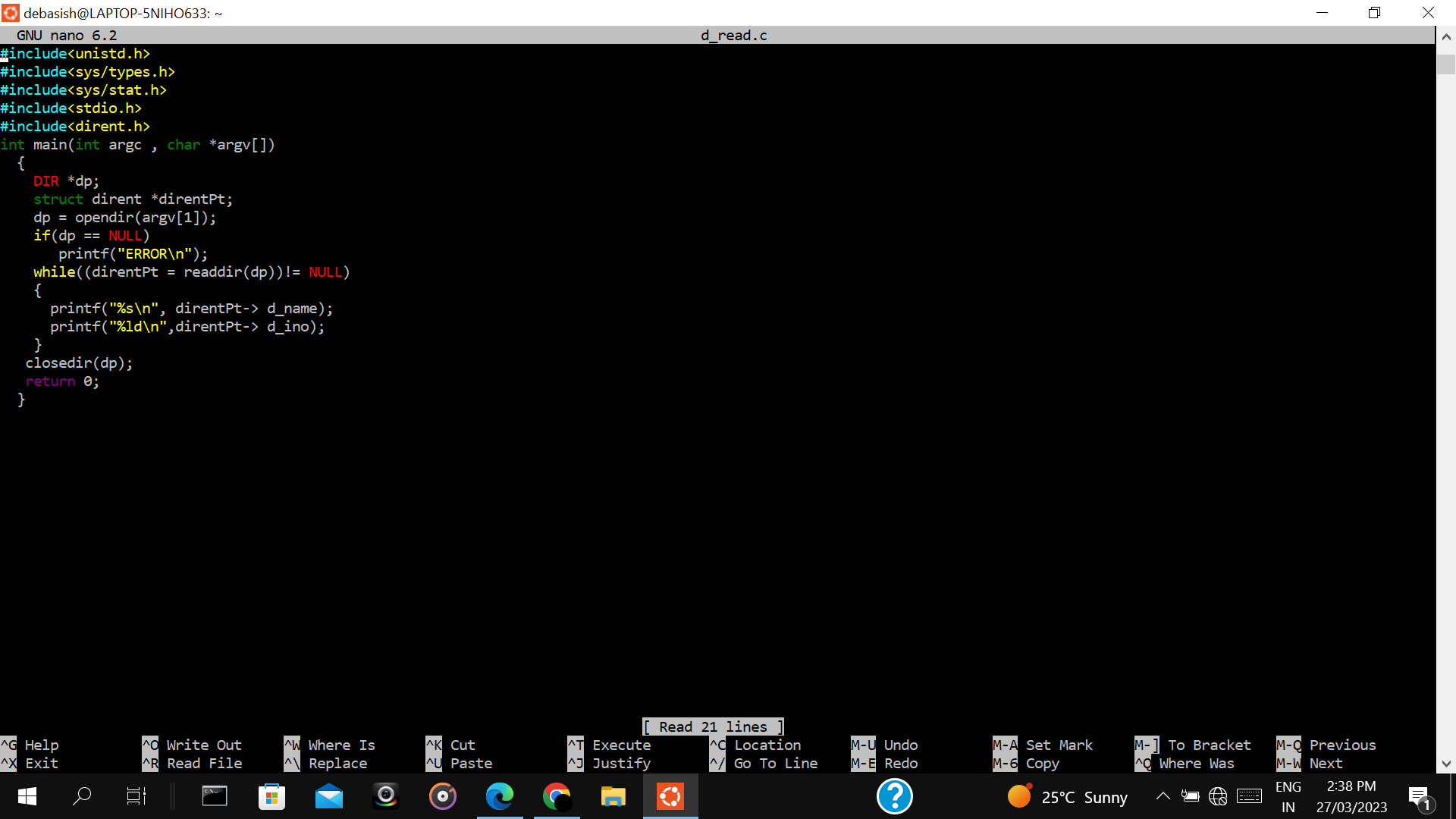


**Experiment 4:**

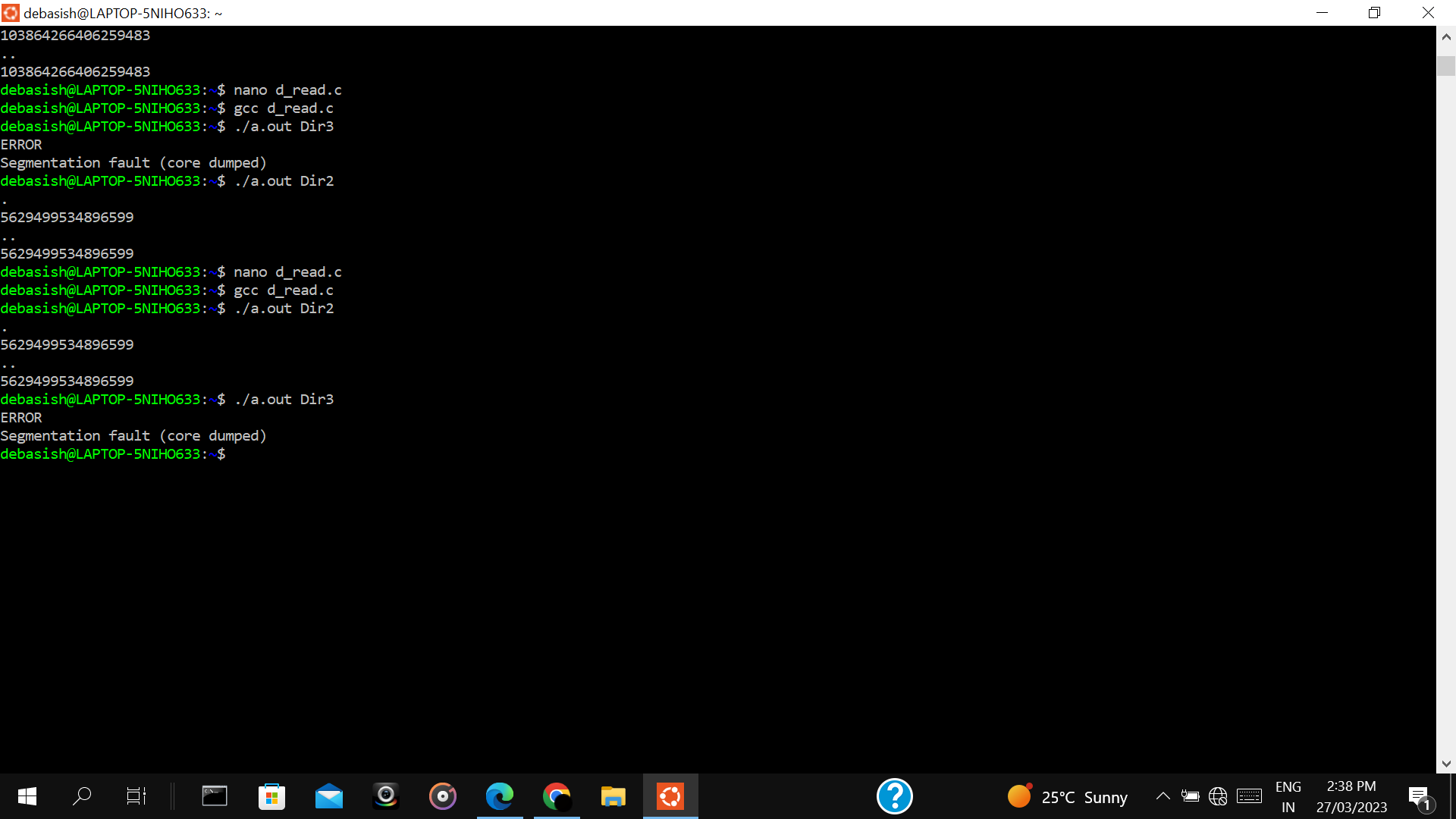
**d\_create.c:-**



**d\_read.c:-**



**Read directory(uses opendir):-**



$ nano d\_readdir.c

# include<uinstd.h>

#include<sys/types.h>

#include<sys.stat.h>

#include<stdio.h>

#include<dirent.h>

Int main(){

DIR \*op

Struct dirent \*direntPt;

dp = opendir(“PSC/Dir1”);

if(dp == NULL)

printf(“error\n”);

while((direntPt = readdir(dp) != NULL){

printf(“%S\n”, direntPt -> d\_name);

printf(“%ld\n”, direntPt -> d\_ino);

}

closedir(dp);

return 0;

}

Compile

gcc d\_readdir.c

Run

$ ./a.out PSC