Date 11/9/25

Expt. No. 4

Page No. 29

EXPERIMENT NO. : 4

Variable and Scope of Variable

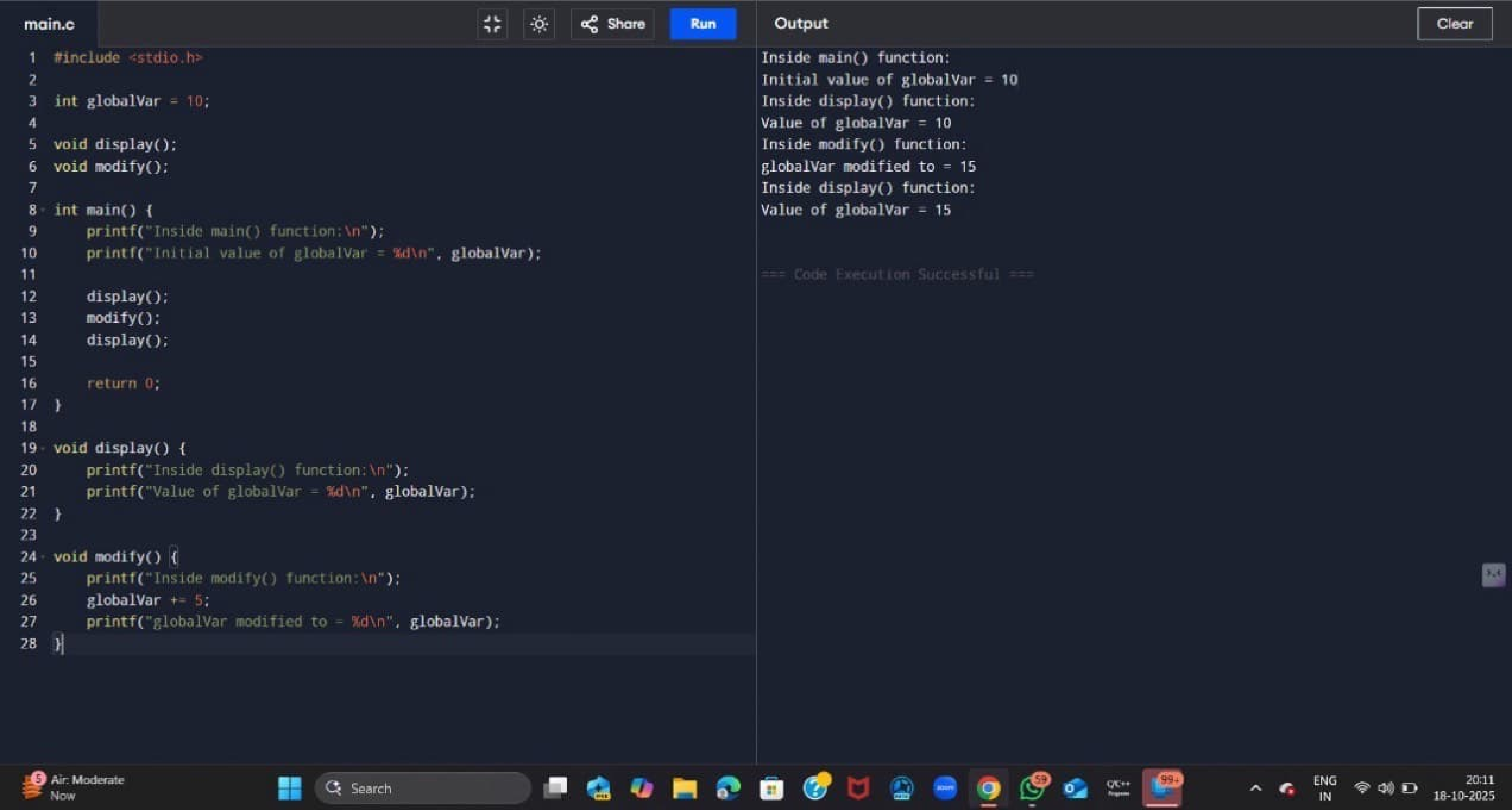
1). Declare a global variable outside all functions and use it inside various functions to understand its accessibility.

```
#include <stdio.h>
int globalVar = 50;

void showValue ();
void changeValue ();
void againShowValue ();

int main ()
{
 printf ("Inside main (): globalVar = %d\n",
                                        globalVar);

  showValue ();
  changeValue ();
  againShowValue ();

  return 0;
 }

void showValue () {
    printf ("Inside showValue (): globalVar = %d\n",
                                        globalVar);
```

Teacher's Signature

```
}
void changeValue () {
    globalVar = globalVar + 25 ;
    printf ("Inside changeValue (): globalVar
                    updated to %d\n", globalVar);
}
void againShowValue () {
    printf ("Inside againShowValue (): globalVar = %d\n",
                                        globalVar);
}
```

Output :-
```
Inside main (): globalVar = 50

Inside showValue (): globalVar = 50

Inside changeValue (): globalVar updated
                         to 75

Inside againShowValue (): globalVar = 75
```



```c
#include <stdio.h>

int globalVar = 10;

void display();
void modify();

int main() {
    printf("Inside main() function:\n");
    printf("Initial value of globalVar = %d\n", globalVar);

    display();
    modify();
    display();

    return 0;
}

void display() {
    printf("Inside display() function:\n");
    printf("Value of globalVar = %d\n", globalVar);
}

void modify() {
    printf("Inside modify() function:\n");
    globalVar += 5;
    printf("globalVar modified to = %d\n", globalVar);
}
```

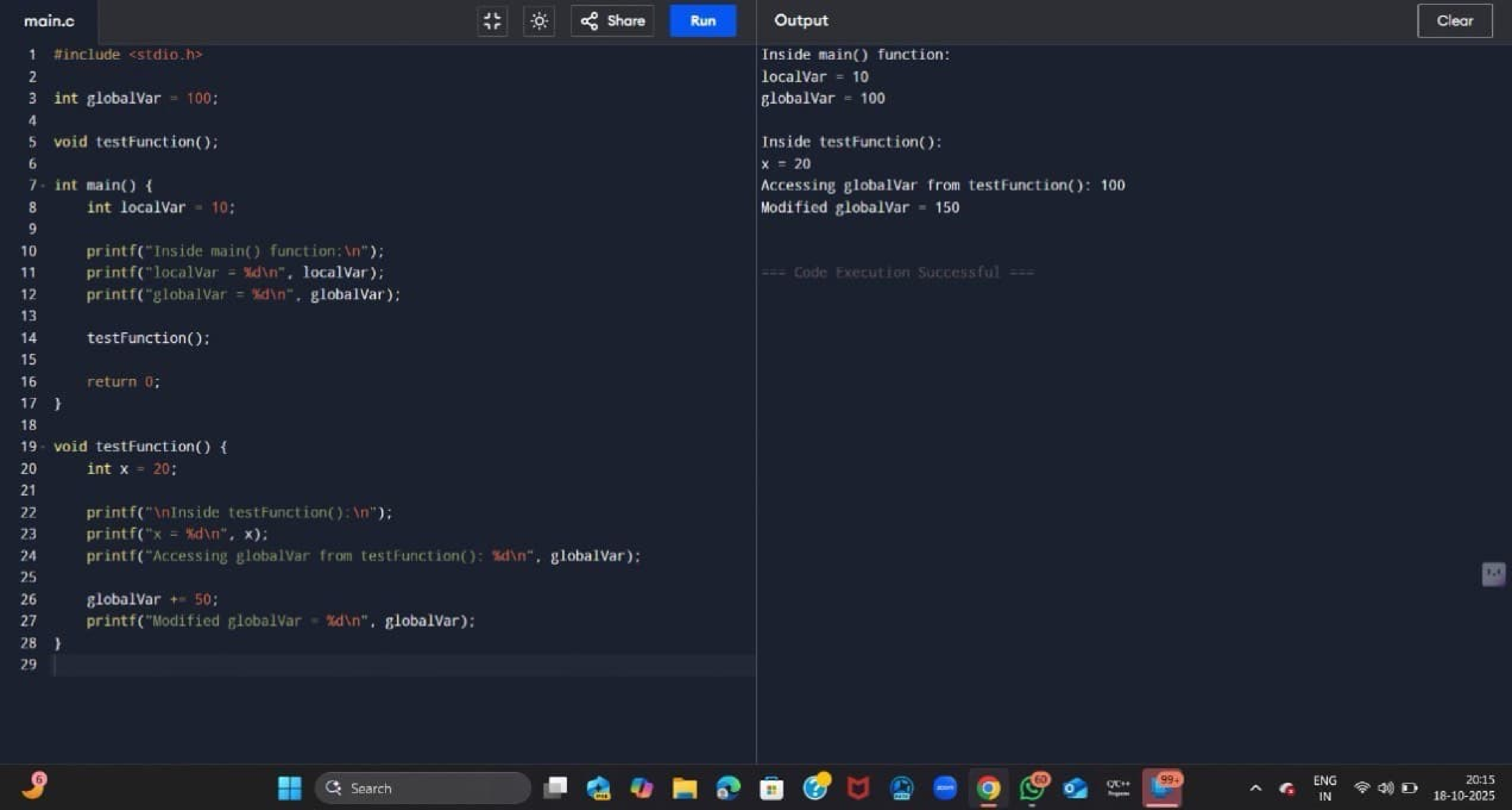
```
Inside main() function:
Initial value of globalVar = 10
Inside display() function:
Value of globalVar = 10
Inside modify() function:
globalVar modified to = 15
Inside display() function:
Value of globalVar = 15


=== Code Execution Successful ===
```

2). Declare a local variable inside a function and try to access it outside the function. Compare this with accessing the global variable from within the function.

```
#include <stdio.h>
int globalVar = 50;

void displayFunction () {
 int localVar = 10;
 printf ("Inside displayFunction :\n");
 printf ("  localVar = %d\n", localVar);
 printf ("  globalVar = %d\n", globalVar);
}
int main () {
  displayFunction ();
  printf ("\n Inside main: \n");
  printf ("  globalVar = %d \n", global Var);
  return 0;
}
```

Output :-

```
Inside displayFunction :
  localVar = 10
  global Var = 50

Inside main :
   global Var = 50
```



```c
#include <stdio.h>

int globalVar = 100;

void testFunction();

int main() {
    int localVar = 10;

    printf("Inside main() function:\n");
    printf("localVar = %d\n", localVar);
    printf("globalVar = %d\n", globalVar);

    testFunction();

    return 0;
}

void testFunction() {
    int x = 20;

    printf("\nInside testFunction():\n");
    printf("x = %d\n", x);
    printf("Accessing globalVar from testFunction(): %d\n", globalVar);

    globalVar += 50;
    printf("Modified globalVar = %d\n", globalVar);
}
```

```
Inside main() function:
localVar = 10
globalVar = 100

Inside testFunction():
x = 20
Accessing globalVar from testFunction(): 100
Modified globalVar = 150


=== Code Execution Successful ===
```

Date

Expt. No.

Page No. 32

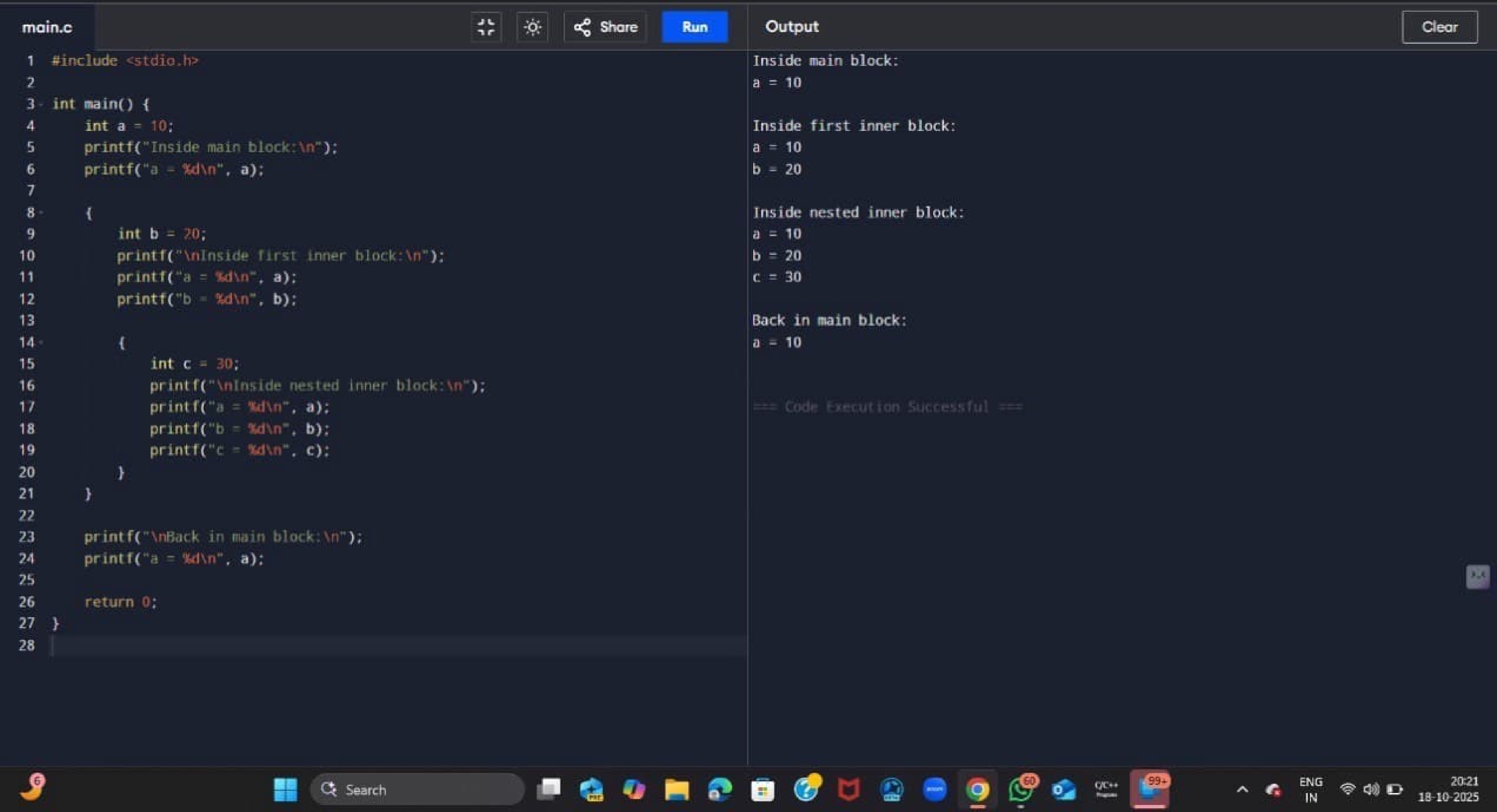
3). Declare variables within different code blocks (enclosed by curly braces) and test their accessibility within and outside those blocks.

```
#include <stdio.h>
int main ( )
{
int x = 5
printf ("In main block: x = %d\n", x);

{
 int y = 10;
 printf ("In first inner block: x = %d, y= %d\n",
                                           x, y);
 {
   int z = 15;
   printf (" In nested block: x = %d, y = %d,
                              z = %d\n", x, y, z);
   }
  }
 printf ("Back in main block: x = %d\n", x);

 return 0;
}
```

Output :-

```
In main block: x = 5
In first inner block: x = 5 , y = 10
In nested block: x = 5 , y = 10 , z = 15
Back in main block: x = 5
```

Teacher's Signature



```c
#include <stdio.h>

int main() {
    int a = 10;
    printf("Inside main block:\n");
    printf("a = %d\n", a);

    {
        int b = 20;
        printf("\nInside first inner block:\n");
        printf("a = %d\n", a);
        printf("b = %d\n", b);

        {
            int c = 30;
            printf("\nInside nested inner block:\n");
            printf("a = %d\n", a);
            printf("b = %d\n", b);
            printf("c = %d\n", c);
        }
    }

    printf("\nBack in main block:\n");
    printf("a = %d\n", a);

    return 0;
}
```

```
Inside main block:
a = 10

Inside first inner block:
a = 10
b = 20

Inside nested inner block:
a = 10
b = 20
c = 30

Back in main block:
a = 10


=== Code Execution Successful ===
```

Date

Expt. No.

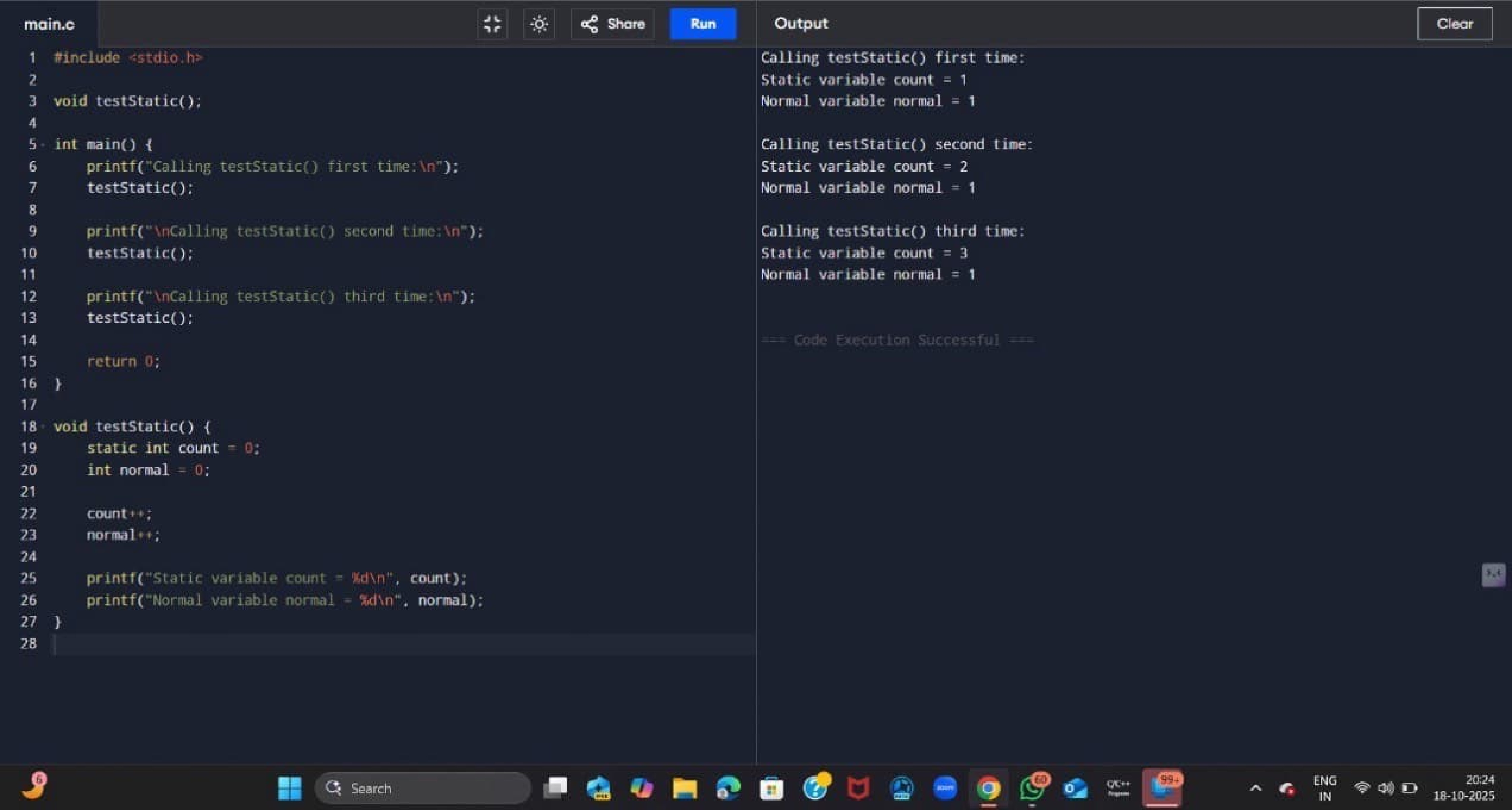
Page No. 33

4) Declare a static local variable inside a function. Observe how its value persists across function calls.

```
# include   <stdio.h>

void  test Static ( )
{
static int counter = 0 ;
counter ++ ;
printf (" counter = %d n" , counter) ;
}

int main ( ) {
     printf ("Calling test Static multiple times : \n");

     test Statics ( ) ;
     test Statics ( ) ;
     test Statics ( ) ;
     test Statics ( ) ;

 return 0 ;
}
```

Output :-
```
Calling test Statics mu
counter = 1
counter = 2
counter = 3
counter = 4
```

Teacher's Signature



```c
#include <stdio.h>

void testStatic();

int main() {
    printf("Calling testStatic() first time:\n");
    testStatic();

    printf("\nCalling testStatic() second time:\n");
    testStatic();

    printf("\nCalling testStatic() third time:\n");
    testStatic();

    return 0;
}

void testStatic() {
    static int count = 0;
    int normal = 0;

    count++;
    normal++;

    printf("Static variable count = %d\n", count);
    printf("Normal variable normal = %d\n", normal);
}
```

```
Calling testStatic() first time:
Static variable count = 1
Normal variable normal = 1

Calling testStatic() second time:
Static variable count = 2
Normal variable normal = 1

Calling testStatic() third time:
Static variable count = 3
Normal variable normal = 1


=== Code Execution Successful ===
```