EXPERIMENT : 8

POINTERS

1). Declare different types of pointers (int, float, char) and initialize them with the address of variables. Print the values of both the pointers and the variables they point to.

```
#include <stdio.h>
int main () {
        int a = 10 ;
        float b = 25.5 ;
        char c = 'A' ;

        int *ptr1 = &a;
        float *ptr2 = &b;
        char *ptr3 = &c;

printf ("values of variables :\n");
printf ("a = %d\n", a);
printf (" b= %.2f\n", b);
printf (" c = %c\n\n", c);

printf ("Values using pointers :\n");
printf ("*ptr1 = %d\n", *ptr1);
printf (" *ptr2 = %2f\n", *ptr2);
```

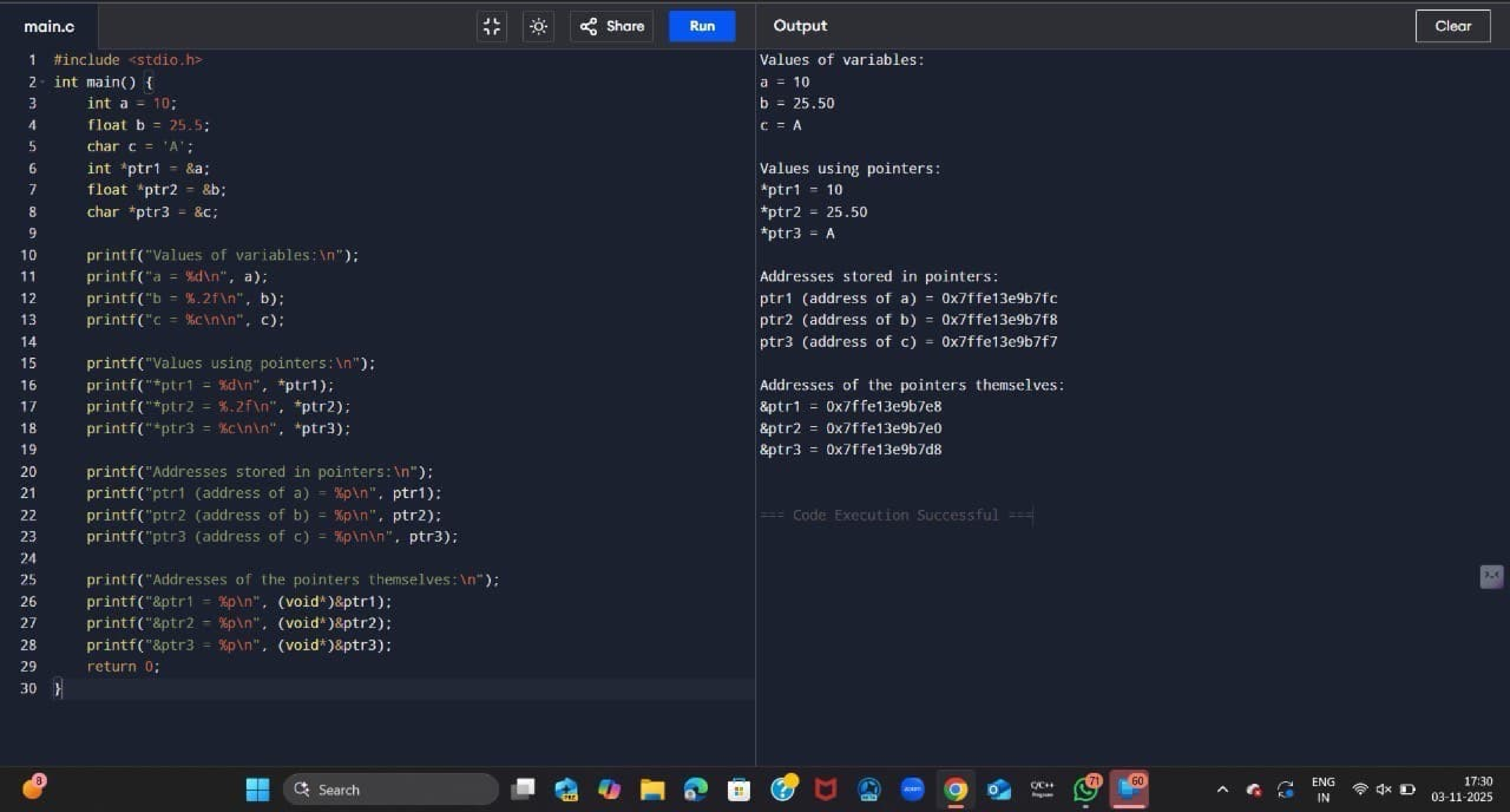
```
printf ("*ptr 3 = %c\n\n", *ptr 3);

printf ("Addresses stored in pointers:\n");
printf ("*ptr1 (address of a) = %p\n", ptr1);
printf ("*ptr 2 (address of b) = %p\n", ptr2);
printf ("*ptr 3 (address of c) = %p\n\n", ptr 3);

printf ("Addresses of the pointers themselves:\n");
printf ("& ptr 1 = %p\n", (void *) & ptr 1);
printf ("& ptr2 = %p\n", (void *) & ptr 2);
printf ("& ptr 3 = %p\n", (void *) & ptr 3);
return 0;
}
```

Output :-

```
Values of variables :
    a = 10 ,          b = 25.50 ,        c = A
  Values using pointers:
   *ptr 1 = 10 ,      *ptr 2 = 25.50 ,    *ptr 3 = A
  Addresses stored in pointers:
  ptr 1 (address of a) = 0x7ffe13e9b7fc
  ptr 2 (address of b) = 0x7ffe13e9b7f8
  ptr 3 (address of c) = 0x7ffe13e9b7f7
  Addresses of the pointers themselves :
  & ptr 1 = 0x7ffe13e9b7e8
  & ptr 2 = 0x7ffe13e9b7e0
  & ptr 3 = 0x7ffe13e9b7d8
```



```c
#include <stdio.h>
int main() {
    int a = 10;
    float b = 25.5;
    char c = 'A';
    int *ptr1 = &a;
    float *ptr2 = &b;
    char *ptr3 = &c;

    printf("Values of variables:\n");
    printf("a = %d\n", a);
    printf("b = %.2f\n", b);
    printf("c = %c\n\n", c);

    printf("Values using pointers:\n");
    printf("*ptr1 = %d\n", *ptr1);
    printf("*ptr2 = %.2f\n", *ptr2);
    printf("*ptr3 = %c\n\n", *ptr3);

    printf("Addresses stored in pointers:\n");
    printf("ptr1 (address of a) = %p\n", ptr1);
    printf("ptr2 (address of b) = %p\n", ptr2);
    printf("ptr3 (address of c) = %p\n\n", ptr3);

    printf("Addresses of the pointers themselves:\n");
    printf("&ptr1 = %p\n", (void*)&ptr1);
    printf("&ptr2 = %p\n", (void*)&ptr2);
    printf("&ptr3 = %p\n", (void*)&ptr3);
    return 0;
}
```

```
Values of variables:
a = 10
b = 25.50
c = A

Values using pointers:
*ptr1 = 10
*ptr2 = 25.50
*ptr3 = A

Addresses stored in pointers:
ptr1 (address of a) = 0x7ffe13e9b7fc
ptr2 (address of b) = 0x7ffe13e9b7f8
ptr3 (address of c) = 0x7ffe13e9b7f7

Addresses of the pointers themselves:
&ptr1 = 0x7ffe13e9b7e8
&ptr2 = 0x7ffe13e9b7e0
&ptr3 = 0x7ffe13e9b7d8


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

2) Perform pointer arithmetic (increment and decrement) on pointers of different data types Observe how the memory addresses change & the effects on data access

```
#include <stdio.h>
int main () {
    int a[3] = {10, 20, 30};
    float b[3] = {1.1, 2.2, 3.3};
    char c[3] = {'A', 'B', 'C'};

    int *pInt = a;
    float *pFloat = b;
    char *pChar = c;

    printf("Initial addresses and values: \n");
    printf("pInt = %p, *pInt = %d\n", (void*)
                        pInt, *pInt);
    printf("pFloat = %p, *pFloat = %.1f\n", (void*)
                        pFloat, *pFloat);
    printf("pChar = %p, *pChar = %c\n\n", (void*)
                        pChar, *pChar);

   pInt++;
   pFloat++;
   pChar++;

   printf("After incrementing pointers: \n");
```

```
    printf ("pInt = %pInt = %d\n", (void*)pInt,
                                          *pInt);
    printf ("pFloat= %pFloat = %.1f\n", (void*)pFloat,
                                          *pFloat);
    printf ("pChar = %pChar = %c\n\n", (void *) pChar,
                                          *pChar);

    pInt --;
    pFloat --;
    pChar --;

    printf ("After decrementing pointers (back to
             original positions): \n");
    printf ("pInt = %p, *pInt = %d\n", (void*)
                              pInt, *pInt);
    printf ("pFloat = %p, *pFloat = %.1f\n", (void *)
                              pFloat, *pFloat);
    printf ("pChar = %p, *pChar= %c\n\n", (void *)
                              pChar, *pChar);
    return 0;
}
```

Output :-

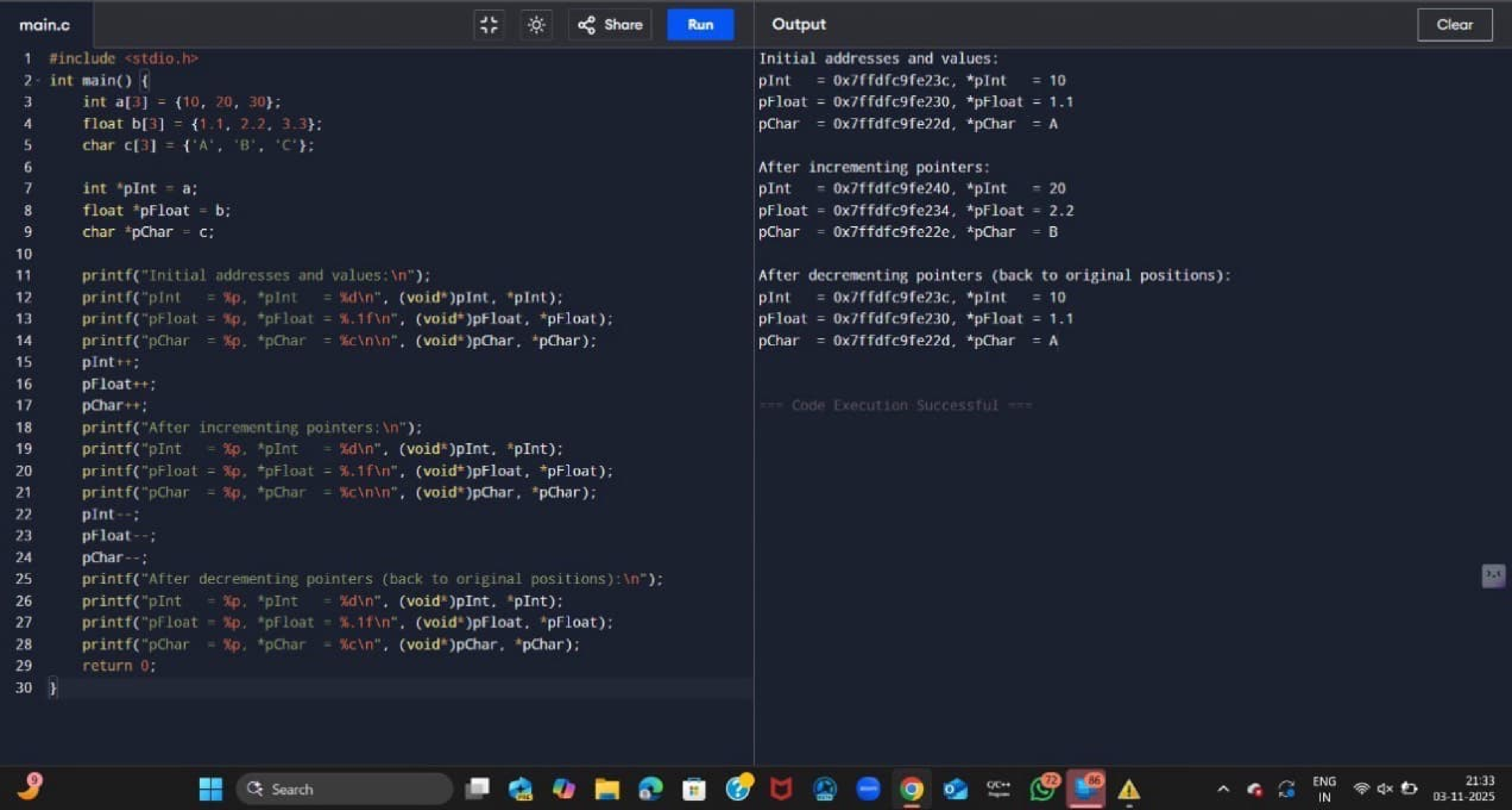
```
Initial addresses and values :

pInt = 0x7ffdfc9fe23c , *pInt = 10
pFloat = 0x7ffdfc9fe230 , *pFloat = 1.1
pChar = 0x7ffdfc9fe22d , *pChar = A

After incrementing pointers :

pInt = 0x7ffdfc9fe240 , *pInt = 20
pFloat = 0x7ffdfc9fe234 , *pFloat = 2.2
pChar = 0x7ffdfc9fe22e , *pChar = B

After decrementing pointers (back to original
 positions ):
pInt = 0x7ffdfc9fe23c , *pInt = 10
pFloat = 0x7ffdfc9fe230 , *pFloat = 1.1
pChar = 0x7ffdfc9fe22d , *pChar = A
```



```c
#include <stdio.h>
int main() {
    int a[3] = {10, 20, 30};
    float b[3] = {1.1, 2.2, 3.3};
    char c[3] = {'A', 'B', 'C'};

    int *pInt = a;
    float *pFloat = b;
    char *pChar = c;

    printf("Initial addresses and values:\n");
    printf("pInt   = %p, *pInt   = %d\n", (void*)pInt, *pInt);
    printf("pFloat = %p, *pFloat = %.1f\n", (void*)pFloat, *pFloat);
    printf("pChar  = %p, *pChar  = %c\n\n", (void*)pChar, *pChar);
    pInt++;
    pFloat++;
    pChar++;
    printf("After incrementing pointers:\n");
    printf("pInt   = %p, *pInt   = %d\n", (void*)pInt, *pInt);
    printf("pFloat = %p, *pFloat = %.1f\n", (void*)pFloat, *pFloat);
    printf("pChar  = %p, *pChar  = %c\n\n", (void*)pChar, *pChar);
    pInt--;
    pFloat--;
    pChar--;
    printf("After decrementing pointers (back to original positions):\n");
    printf("pInt   = %p, *pInt   = %d\n", (void*)pInt, *pInt);
    printf("pFloat = %p, *pFloat = %.1f\n", (void*)pFloat, *pFloat);
    printf("pChar  = %p, *pChar  = %c\n", (void*)pChar, *pChar);
    return 0;
}
```

```
Initial addresses and values:
pInt   = 0x7ffdfc9fe23c, *pInt   = 10
pFloat = 0x7ffdfc9fe230, *pFloat = 1.1
pChar  = 0x7ffdfc9fe22d, *pChar  = A

After incrementing pointers:
pInt   = 0x7ffdfc9fe240, *pInt   = 20
pFloat = 0x7ffdfc9fe234, *pFloat = 2.2
pChar  = 0x7ffdfc9fe22e, *pChar  = B

After decrementing pointers (back to original positions):
pInt   = 0x7ffdfc9fe23c, *pInt   = 10
pFloat = 0x7ffdfc9fe230, *pFloat = 1.1
pChar  = 0x7ffdfc9fe22d, *pChar  = A


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

Experiment No.________ Name:________

PAGE NO.:

DATE: / /

3). Write a function that accept pointers as parameters. Pass variables by reference using pointers and modify their values within the function

```
#include <stdio.h>

void modifyValues (int *x, float *y, char *z){
    *x = *x + 10;
    *y = *y + 2;
    *z = *z + 1;
}
int main () {
        int a = 5;
        float b = 3.5;
        char c = 'A';

        printf ("Before modification : \n");
        printf ("a = %d\n", a);
        printf ("b = %.2f\n", b);
        printf ("c = %c\n\n", c);

  modifyValues (&a, &b, &c);

       printf ("After modification (inside function):\n");
       printf ("a = %d\n", a);
       printf ("b = %.2f\n", b);
       printf ("c = %c\n\n, c);
```

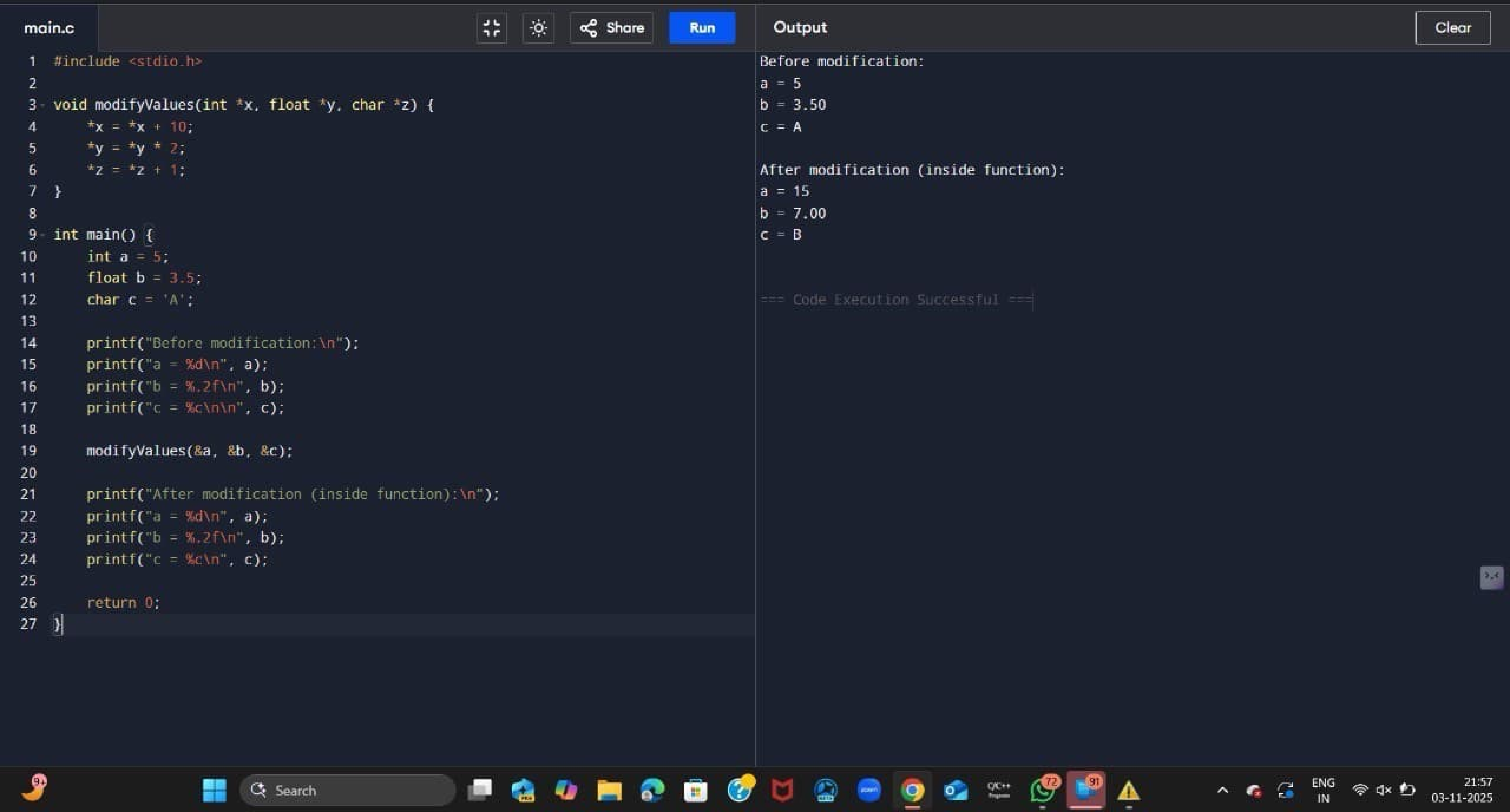
Teacher's Signature:________

```
        return 0;
}
```

Output :-

```
Before modification :
a = 5
b=3.50
c = A

After modification (inside function):
a =15
b=7.00
C = B
```



```c
#include <stdio.h>

void modifyValues(int *x, float *y, char *z) {
    *x = *x + 10;
    *y = *y * 2;
    *z = *z + 1;
}

int main() {
    int a = 5;
    float b = 3.5;
    char c = 'A';

    printf("Before modification:\n");
    printf("a = %d\n", a);
    printf("b = %.2f\n", b);
    printf("c = %c\n\n", c);

    modifyValues(&a, &b, &c);

    printf("After modification (inside function):\n");
    printf("a = %d\n", a);
    printf("b = %.2f\n", b);
    printf("c = %c\n", c);

    return 0;
}
```

```
Before modification:
a = 5
b = 3.50
c = A

After modification (inside function):
a = 15
b = 7.00
c = B


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