Title: Pointers in C.

Objective:

The main objectives of this lab are to

- Learn about Pointer Declaration & Arithmatic Operations (increament/decreament) using pointers
- Learn about how to Swap two number using pointers
- Learn about how to Array address in 1D using pointer

Theory:

A **pointer** is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before using it to store any variable address. The general form of a pointer variable declaration is –

```
type *var-name;
```

Here, **type** is the pointer's base type; it must be a valid C data type and **var-name** is the name of the pointer variable. The asterisk * used to declare a pointer is the same asterisk used for multiplication. However, in this statement the asterisk is being used to designate a variable as a pointer. Take a look at some of the valid pointer declarations –

```
int *ip; /* pointer to an integer */
double *dp; /* pointer to a double */
float *fp; /* pointer to a float */
char *ch /* pointer to a character */
```

The actual data type of the value of all pointers, whether integer, float, character, or otherwise, is the same, a long hexadecimal number that represents a memory address. The only difference between pointers of different data types is the data type of the variable or constant that the pointer points to.

NULL Pointers

It is always a good practice to assign a NULL value to a pointer variable in case you do not have an exact address to be assigned. This is done at the time of variable declaration. A pointer that is assigned NULL is called a null pointer.

Assigning addresses to Pointers

Let's take an example.

```
int* pc, c;
c = 5;
pc = &c;
```

Here, 5 is assigned to the c variable. And, the address of c is assigned to the pc poi

Get Value of Thing Pointed by Pointers

To get the value of the thing pointed by the pointers, we use the * operator. For example:

```
int* pc, c;
c = 5;
pc = &c;
printf("%d", *pc);   // Output: 5
```

Here, the address of c is assigned to the pc pointer. To get the value stored in that address, we used *pc.

Changing Value Pointed by Pointers

Let's take an example.

```
int* pc, c;
c = 5;
pc = &c;
c = 1;
printf("%d", c);    // Output: 1
printf("%d", *pc);  // Ouptut: 1
```

We have assigned the address of c to the pc pointer.

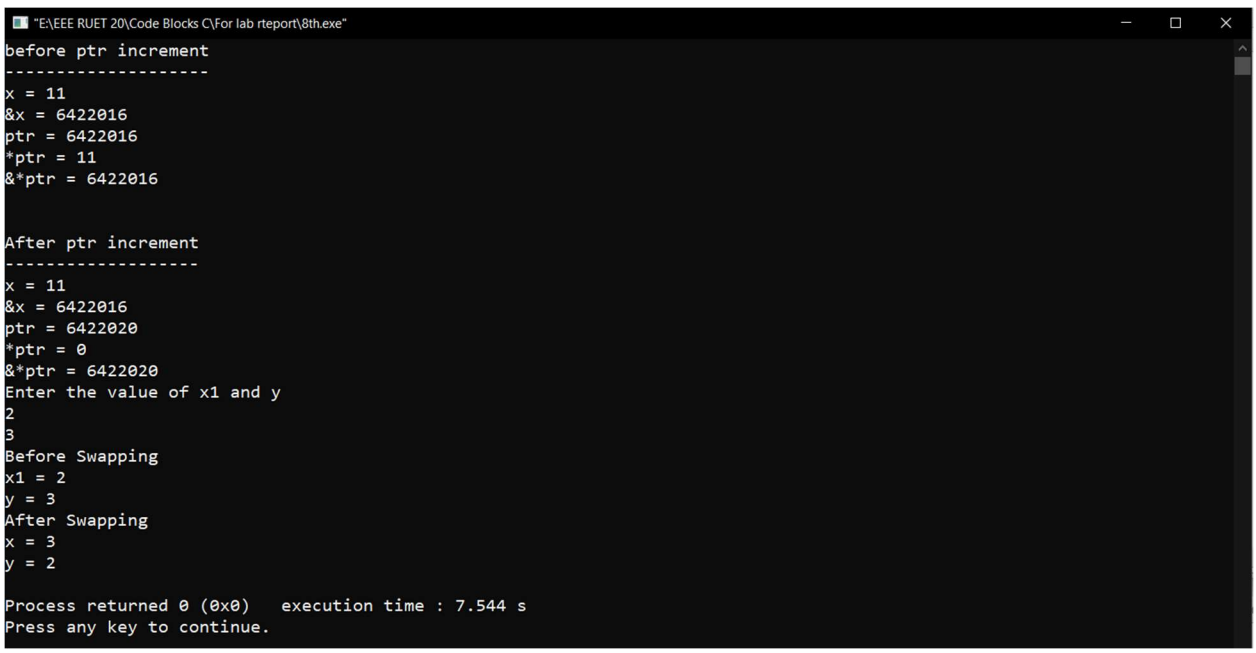
Then, we changed the value of c to 1. Since pc and the address of c is the same, *pc gives us 1.

Source Code:

```
/// Pointer Declaration & Arithmatic Operations (increament/decreament).
#include <stdio.h>
int main()
{
    int x; // Pointer Declaration
    int *ptr;

    x = 10;
    ptr = &x;
        *ptr = *ptr + 1;
        // Pointer Increment/Decrement
        printf("before ptr increment\n");
        printf("--------------------\n");
        printf("x = %d\n", x);
        printf("&x = %d\n", &x);
        printf("ptr = %d\n", ptr);
        printf("*ptr = %d\n", *ptr);
        printf("&*ptr = %d\n\n\n", &*ptr);

        ptr = ptr +1;
        printf("After ptr increment\n");
        printf("-------------------\n");
        printf("x = %d\n", x);
        printf("&x = %d\n", &x);
        printf("ptr = %d\n", ptr);
        printf("*ptr = %d\n", *ptr);
        printf("&*ptr = %d\n", &*ptr);

    /// Swap two number using pointers.

        int x1, y, *a, *b, temp;

        printf("Enter the value of x1 and y\n");
        scanf("%d%d", &x1, &y);

        printf("Before Swapping\nx1 = %d\ny = %d\n", x1, y);

        a = &x1;
        b = &y;

```

```
        temp = *b;
        *b = *a;
        *a = temp;

        printf("After Swapping\nx = %d\ny = %d\n", x1, y);
        return 0;
    }
```

Output:



Discussion and Conclusion:

In this program, I work on pointers. In the first program I done arithmetic operations – Increment/Decrement. I also show that how to declare pointer within the same program. In the next program I made a program that takes input 2 numbers from the user and display those numbers by swapping.