

Lab No: 1

Date: 2081/11/20

**A. Write a program in C for finding the GCD of two numbers.**

---

**Theory:**

The Greatest Common Divisor (GCD) of two numbers is the largest number that divides both without leaving a remainder. Euclidean algorithm is the most effective way to find the GCD. GCD can be found using the Euclidean Algorithm as:

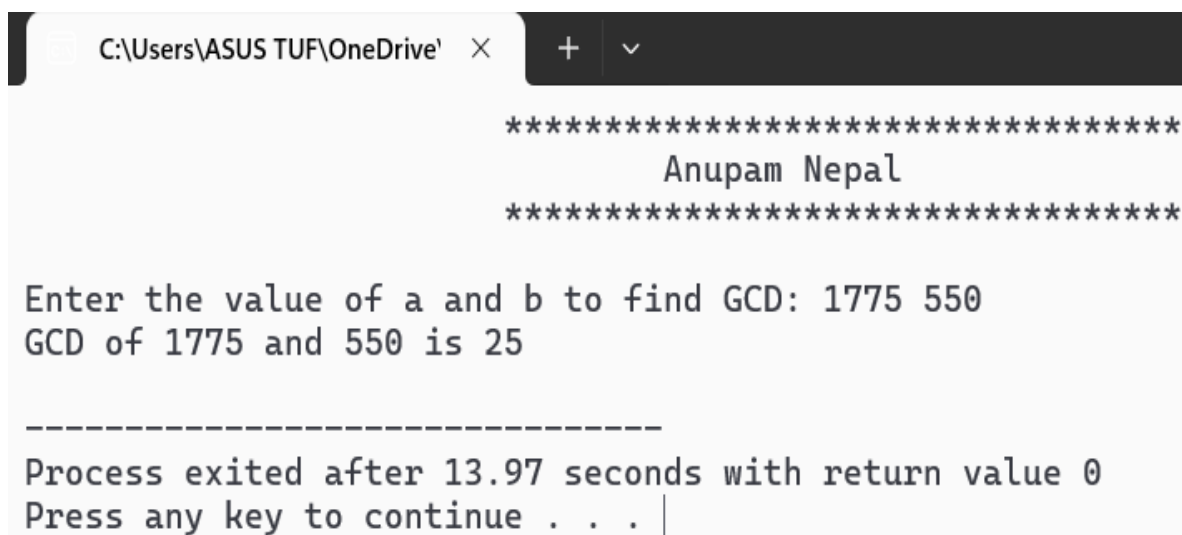
$$\text{GCD}(a, b) = \text{GCD}(b, a \bmod b), \text{ until } b=0.$$

**Source Code:**

```
#include <stdio.h>
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

int main() {
    printf("Enter the value of a and b to find GCD: ");
    scanf("%d %d",&a,&b);
    printf("GCD of %d and %d is %d\n", a, b, gcd(a, b));
    return 0;
}
```

**OUTPUT:**



```
C:\Users\ASUS TUF\OneDrive' X + v
*****
Anupam Nepal
*****

Enter the value of a and b to find GCD: 1775 550
GCD of 1775 and 550 is 25

-----
Process exited after 13.97 seconds with return value 0
Press any key to continue . . . |
```

## B. Write a program in C for printing the Fibonacci Series upto n<sup>th</sup> term.

---

### Theory:

Fibonacci Series is a sequence of numbers where each number is the sum of the two preceding ones. It starts with 0 and 1.

### Formula:

$$F(n)=F(n-1) + F(n-2)$$

where:

- $F(0)=0$
- $F(1)=1$

### Source Code:

```
#include <stdio.h>

void fibonacci(int n) {
    int first = 0, second = 1, i, next;

    printf("Fibonacci Series: %d %d ", first, second);

    for (i = 2; i < n; i++) {
        next = first + second;

        printf("%d ", next);

        first = second;
        second = next;
    }

    printf("\n");
}

int main() {
    int n;

    printf("Enter the number of terms: ");

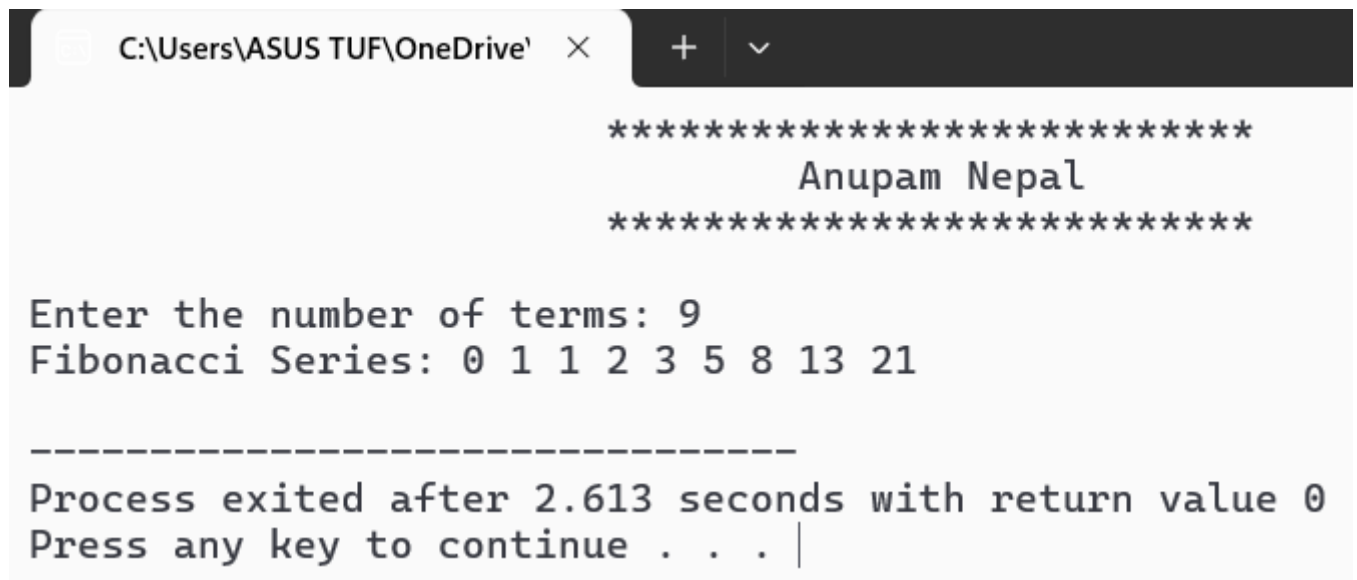
    scanf("%d", &n);

    if (n < 1) {

        printf("Please enter a positive integer.\n");
    }
}
```

```
} else if (n == 1) {  
  
    printf("Fibonacci Series: 0\n");  
  
} else {  
  
    fibonacci(n);  
  
}  
  
return 0;  
  
}
```

### OUTPUT:



```
C:\Users\ASUS TUF\OneDrive' X + v  
  
*****  
Anupam Nepal  
*****  
  
Enter the number of terms: 9  
Fibonacci Series: 0 1 1 2 3 5 8 13 21  
  
-----  
Process exited after 2.613 seconds with return value 0  
Press any key to continue . . . |
```

## C. Write a program in C implementing Bubble Sort.

---

### Theory:

Bubble Sort is a simple sorting algorithm that repeatedly swaps adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity are quite high.

**Time Complexity:**  $O(n^2)$

### Working

1. Compare adjacent elements.
2. Swap if the first is greater than the second.
3. Repeat until the list is sorted.

### Source Code:

```
#include <stdio.h>

void bubble_sort(int arr[], int n) {
    int i,j;

    for (i = 0; i < n - 1; i++) {
        int swapped = 0;

        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                swapped = 1;
            }
        }

        if (!swapped) break;
    }
}

void print_array(int arr[], int n) {
    int i;

    for (i = 0; i < n; i++)
```

```
        printf("%d ", arr[i]);

    printf("\n");
}

int main() {

    int arr[] = {5, 33, 18, 44, 22, 56, 23};

    int n = sizeof(arr) / sizeof(arr[0]);

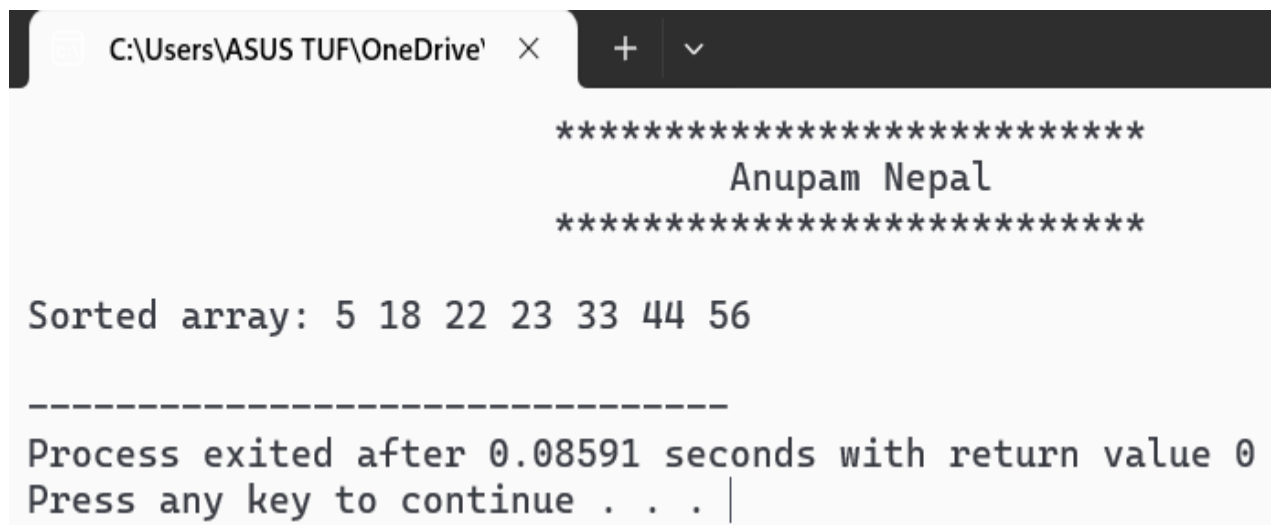
    bubble_sort(arr, n);

    printf("Sorted array: ");

    print_array(arr, n);

    return 0;
}
```

### OUTPUT:



```
C:\Users\ASUS TUF\OneDrive' X + v

*****
Anupam Nepal
*****

Sorted array: 5 18 22 23 33 44 56

-----
Process exited after 0.08591 seconds with return value 0
Press any key to continue . . . |
```

## D. Write a program in C implementing Insertion Sort.

---

### Theory:

Insertion Sort is a sorting algorithm that builds a sorted list one element at a time by inserting each element in its correct position.

**Time Complexity:  $O(n^2)$**

### Working:

- Start with the second element as the first element is assumed to be sorted.
- Compare it with previous elements and swap them if second element is smaller.
- Move to the third element, compare it with the first two elements, and put it in its correct position
- Repeat until the entire array is sorted.

### Source Code:

```
#include <stdio.h>

void insertion_sort(int arr[], int n) {
    int i,j;
    for (i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

void print_array(int arr[], int n) {
    int i;
    for (i = 0; i < n; i++)
```

```
        printf("%d ", arr[i]);

    printf("\n");
}

int main() {

    int arr[] = {2, 25, 17, 31, 43, 37, 12};

    int n = sizeof(arr) / sizeof(arr[0]);

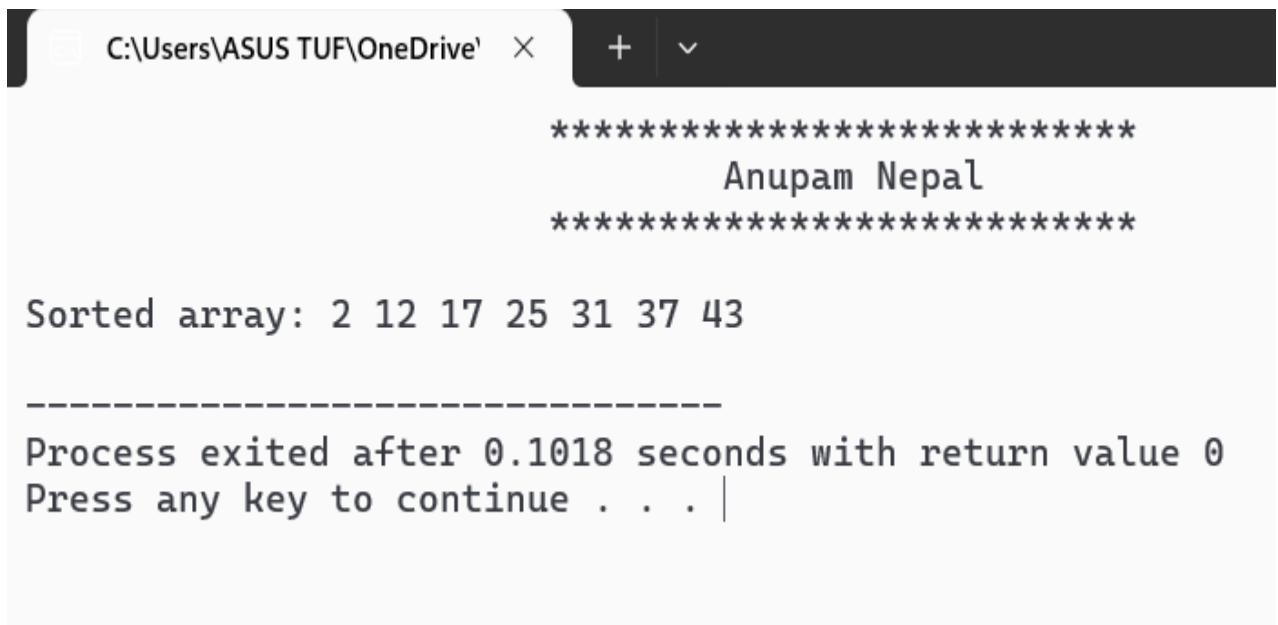
    insertion_sort(arr, n);

    printf("Sorted array: ");

    print_array(arr, n);

    return 0;
}
```

### OUTPUT:



```
C:\Users\ASUS TUF\OneDrive' X + v

*****
Anupam Nepal
*****

Sorted array: 2 12 17 25 31 37 43

-----
Process exited after 0.1018 seconds with return value 0
Press any key to continue . . . |
```

## E. Write a program in C implementing Selection Sort.

---

### Theory:

Selection Sort is a comparison-based sorting algorithm. It sorts an array by repeatedly selecting the smallest (or largest) element from the unsorted portion and swapping it with the first unsorted element. This process continues until the entire array is sorted. It requires less number of swaps (or memory writes) compared to many other standard algorithms.

### Time Complexity: $O(n^2)$

### Working:

- First we find the smallest element and swap it with the first element. This way we get the smallest element at its correct position.
- Then we find the smallest among remaining elements (or second smallest) and swap it with the second element.
- We keep doing this until we get all elements moved to correct position.

### Source Code:

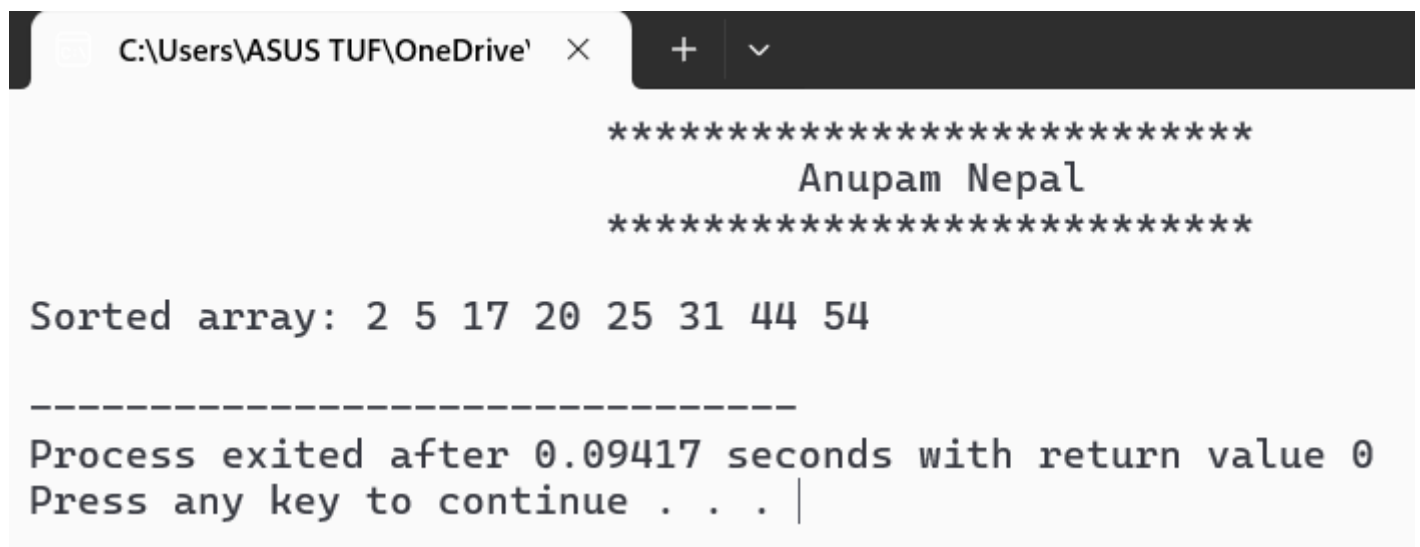
```
#include <stdio.h>
void selection_sort(int arr[], int n)
{
    int i,j;
    for (i = 0; i < n - 1; i++) {
        int min_idx = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
                min_idx = j;
            }
        }
        int temp = arr[i];
        arr[i] = arr[min_idx];
        arr[min_idx] = temp;
    }
}
void print_array(int arr[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main()
{
    int arr[] = {2, 17, 5, 31, 20, 54, 25, 44};
    int n = sizeof(arr) / sizeof(arr[0]);
```



```
selection_sort(arr, n);  
printf("Sorted array: ");  
print_array(arr, n);  
return 0;  
}
```

### OUTPUT:



```
C:\Users\ASUS TUF\OneDrive' X + v  
  
*****  
Anupam Nepal  
*****  
  
Sorted array: 2 5 17 20 25 31 44 54  
  
-----  
Process exited after 0.09417 seconds with return value 0  
Press any key to continue . . . |
```