Assignment - 2.

1] Print elements of an array (forward & backward).

⇒ // Forward direction.

```
Public static void ArrayElements (int arr[], int i){
    if(i == arr.length){
        return;
    }
int length = arr.length;
 if( i >= 4){
    return;
  }
 Print(arr[i]);
 ArrayElements( i+1, arr);
}
```

// backward direction.

```
public static void Reverse_elements (int arr[], int i)
    {   i = arr.length-1;
     if( i<0){
        return;
      }
    Print( arr[i]);
    Reverse_elements ( arr, i-1);
    }
```

⇒ logic ⇒ // forward direction elements

arr[] = {1,2,3,4}

i = 0;

length = 4;

if(0 >= 4) ⇒ false

↓

Print(arr[i] = arr[0])

arr[0] = 1

ArrayElements(1, arr)

i = 1;

if(1 >= 4){

false {

↓

Print(arr[1] = 2)

ArrayElements(2, arr)

i = 2

if(2 >= 4){

false;

↓

Print(arr[2] ⇒ 3

ArrayElements(3, arr)

i = 3;

if(3 >= 4{

false

↓

Print(

arr[3]

⇒ 4

ArrayElements(4, arr);
if(4>=4);
return; → program ends.

// backward direction:

arr[] = {1,2,3,4}
int i = arr.length-1
= 4-1 = 3.
if(3<0);
false
Print(arr[3] => 4)
Reverse_elements(i-1, arr)

int i = 2;
if(2<0);
false
Print(arr[2] => 3
Reverse_elements(i-1, arr)

i = 1;
if(1<0);
false
print(arr[1] => 2)
Reverse_elements(i-1, arr)

i = 0;
if(0<0);
false
Print(arr[0] => 1)
Reverse_elements(i-1, arr)

i = -1,
if(-1<0);
true
return; — ends Here.
}

Q 2] Find the sum of digits of a number.

⇒ algorithm: ⇒ public static int sum_digit(int N, int count);

~~int count = 0;~~

if(N<=0);
return count;
}
int rem;
rem = N % 10;
count = count + rem;
N = N / 10;
Sum_digit(N, count);
}

logic ⇒
N = 1234
count = 0;
if(1234<0)
false
rem = 1234 % 10 ⇒ 4
count = 0 + 4;
count = 4
N = 123
Sum_digit(123, 4);

N = 123
count = 4;
if(123<0);
false
rem = 123 % 10
rem ⇒ 3
count = 4 + 3 = 7
N = 12
Sum_digit(12, 7)

N = 12
count = 7;
if(12<0);
false;
}
rem = 12 % 10 ⇒ 2
count = 7 + 2 ⇒ 9
N = 1
Sum_digit(1, 9)

N = 1
count = 9
if(1<0);
false
}
rem = 1
count = 9 + 1
= 10
N = 0

N = 0
if(0<=0)
{
true
return count;
10

24/04/2025 19:37

Q3) Find the Products of digits of a Number

```
s) public static void Product_of_digit(int num,
                                       int count){
                                          ↓
                                          1
    int rem = 0;
  if( num <=0){
     sprint(count);
     return;
     }
  rem= num%10;
  count= (count*10 + rem
  (ount= (ount+ rem;
   num= num/10;
   Product_of_digit( num, count);
  }.
```

logic :-

num=231, count=1;
if(231<=0){
↓
false
}
rem= 231%10=1
count= 1 * 1;
count=1
num= num/10;
Product_of_digit(23,1)

num=23
count=1
if(23<=0)
↓
false
rem=3
count= 1+3=3
num= num/10
=23/10=2
Product_of_digit(2,3)

num=2
count=3
if(2<=0)
↓
false
↓
}
rem=2
count=3*2=6;
num=2/10=0

num=0,
if(0<=0){
↓
True
Print(6);
return;
}
Product_of_digit(0,6)

Q4) count number of digits in a number.

```
Public static int count_of_digit( int num, count){
                                          ↓
                                          0.
    int rem=0;
   if(num <=0){
      return count;
      }
   rem= num%10;
   count++;
   num= num/10;
   count_of_digit( num, count);
  }
```

N=98765
count=0
if(98765<0
false
↓
rem=5
count=1;

N=9876
count=1
if(9876<0);
false
↓
rem=6
count=2;
num=987

N=987
count=2;
if(987<0)
↓
false
↓
rem=7
count=3;
num=98

N=98
count=3;
if(98<0){
false
rem=8
count=4
num=9
count_of

N=9
count=4
if(9<0){
false
rem=9
count=5;

N=0
count=5
if(0<0)
True
return 5;

5) Find the maximum element in an array.

```
Public static int max_element (int arr[], int i = 3/5
                                          ↓
                                          2
    int max = arr[0];
    if(i == arr.length){
        return max;
    }
    if(arr[i] > max){
        max = arr[i];
    }
    max_element(arr, max, i+1);
```

arr = [2, 5, 9, 1, 6];

logic ⇒ i = 1;

max = arr[0]
max = 2
if(arr[1] = 5 > 5).
↓
False

if(arr[1] > max)
(5 > 2)
↓
True
max = 5;

max_Element(arr, 5, 2)

i = 2
max = 5
if(2 == 5)
↓
False

if(arr[2] > max)
(9 > 5)
↓
True
max = 9

max_element(arr, 9, 3)

i = 3
max = 9
if(3 == 5)
↓
False

if(1 > 9){
↓
False
}
max = 9

max_element(arr, 9, 4)

i = 4
max = 9
if(4 == 5)
↓
False

if(6 > 9)
↓
False
max = 9

max_element(arr, 9, 5)

i = 5
if(5 == 5)
true
return
max = 9

6] check if an array is sorted or not.

⇒ trying logic not perfect

```
public static boolean is_sort(int arr[], int i){
    boolean is_sort_arr = false;
    int temp = arr[i];
    if(i >= arr.length)
    {
        return is_sort_arr;
    }
    if(temp <= arr[i+1]){
        i++;
        is_sort_
```

7) check if number is prime:-

```
public static void is_prime(int num; int i:
                                    );
```

tried logic =>

Not accurate

```
if(num == 1);
    Print(false);
}
if( i >= num/2);
    return;
}
if(num%i==0);
if(i = 1 || i== n);
    if(num%i==0);
        Print("prime)
}
```

8] Find an first index of an element in array

```
Public static void first_index (int arr[], int i,
                                int key) {
    boolean is_find = false;
    if (is_find == true) {
        ~~Print(i)~~
        return;
    }
    if (key == arr[i]) {
        is_find = true;
        Print(i);
    }
    first_index(arr, i+1, key, is_find);
}
```

⇒ logic ⇒

i = 0,
arr = [4, 2, 7, 7, 9]
key = 7
is_find = ~~true~~ false;
if (is_find == true)
↓
false
if (7 == arr[i] ⇒ 4)
↓
false

first_index (arr, 1, 7, false)

i = 1
key = 7
is_find = false;
if (is_find = ~~false~~ true)
↓
false
if (7 == 2)
↓
false

first_index (arr, 2, 7, false)

i = ~~3~~ 2
key = 7
is_find = false;
if (is_find == true)
↓
false
if (7 == 7)
is_find = true;
Print(i) = 2

first_index (arr, 3, 7, true)

i = 3
key = 7
is_find = true;
if (true == true)
↓
true
return

g) Find the last index of an element in an array

```
public static void last_index (int arr[], int i, int key);
int temp=0;  i = 0;
        if ( i == arr.length);
            Print(temp);
            return;
        }
        if (arr[i] == key) {
            temp = i;
        }
last_index ( arr, i+1, key);
```

⇒ logics

i=0
arr = [4, 2, 7, 7, 9] key=7
if(0>=4)
↓
False
if (arr[0] == key)
(4 == 7)
↓
False
temp=0;
}
last_index (arr, 1, 7)

i=1 k<7
if(1>=4)
↓
False
if(2==7)
↓
False
temp=0;
↓
last_index (arr, 2, 7)

i=2
k<7
(2>=4)
↓
False
(7==7)
↓
True
temp=1
temp=2.
last_index(arr, 3, 7)

i=3
k=7
(3>=4)
↓
False
(7==7)
↓
True
temp=1
temp=3;
↓
last_index (arr, 4, 7)

i=4
k<7
(4==4)
↓
True
Print (3)
return;

10] Reverse a Number using Recursion.

```
Public static int reverse_num (int num, int reverse,
                               int i)
    i=0; reverse=0;
    IF{
    ~~while (Num~~
    IF( num <=0){
        return reverse;
        }
    int rem= num %10;
        reverse = reverse * 10 + rem;
        num = num / 10.
    reverse_Num( num, i+1, reverse);
}.
```

logic=>

num=1234
reverse=0;
i=0;
if(1234<=0)
↓
false

rem=num%10
rem=4
~~num=123~~
reverse= 4;
num =123

num=123
reverse=4;
i=1;
if(123<=0)
↓
false

rem = 3
reverse= 4*10+3
=>43
num=12

num=12
reverse=43
i=2;
if(12<=0)
↓
false;
rem=2
reverse= 43*10+2
=432
num=1

num=1
reverse=432
i=3
if(1<=0)
↓
false

rem=1
reverse= 432*10+1
=4321
num=0

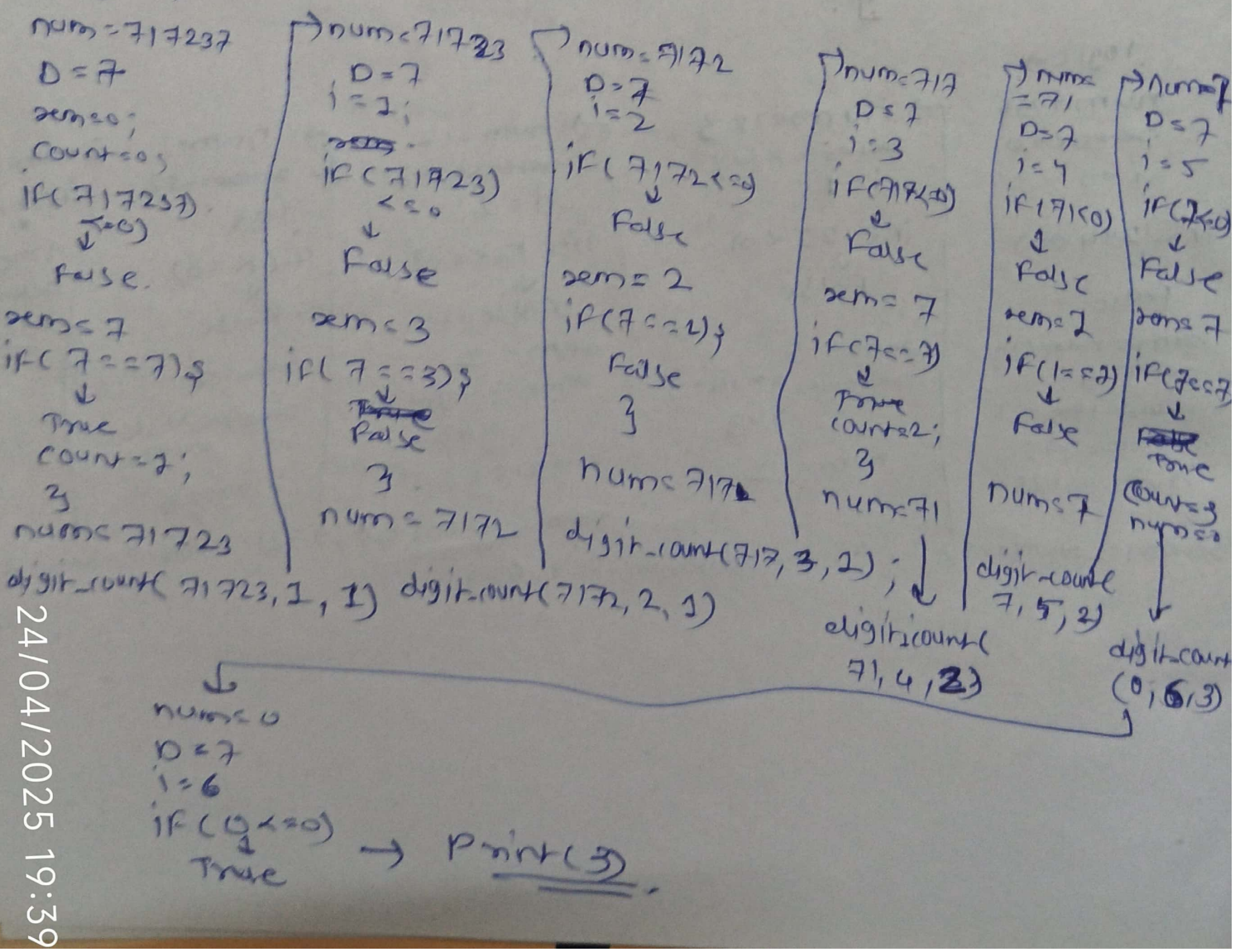
num=0
if(0<=0)
True
return
4321

reverse_Num(123, 1, 4) reverse_Num(12, 2, 43) reverse-Num(1,3,432), reverse_Num(0,4,4321)

24/04/2025 19:38

11) count how many times a digit appear in numbers

```
public static void digit_count(int num, int D, int count);
{
    int rem = 0; count = 0;
    if ( num <=0) {
        Print(count);
    }
    rem = num %10
    if(rem == D) {
        count++;
    }
    num = num/10;
    digit_count(num, D, count);
```



12] check if a number is palindrome or not

```
public static void is_palindrome(int num, int temp_num,
                                 int count) {
    int rem=0;
    count=0;
    if(num<=0) {
        return;
    }
    rem= num%10;
    count= count*10+rem;
    num=num/10;
    is_palindrome(num, temp_num, count);
    if(temp_num==count) {
        return true;
    } else {
        return false;
    }
}

psum() {
    int temp_num=num;
    //copy of original num for checking.
```

logic.

i=0, count=0,
nums 121
temp_num=121
if(12<=0)
↓
false
rem=1
count=0*10+1
count = 1
num=12
is_palindrome(12, 121, 1) 1)

→ i=1,
count=1,
num=12
if(12<=0)
↓
false
rem=2
count=1*10+2
count=>12
num=1

→ i=2
num=1
count=12
if(1<=0)
↓
false
rem=1
count=120+1
=>121
num=0;

→ i=3
num=0
count=121
if(0<=0)
↓
True
return
if(121==121){
return true
}

is_palindrome(1, 121, 2, 12) is_palindrome(0, 121, 3, 121)

19] Find GCD of two numbers using recursion:

```
public static void find_gcd(int n1, int n2,
                            int i, int gcd){
                                   1      0
    int gcd = 0;

    if( i > n1/2){
        print(gcd);
        return;
    }
    if( n1%i==0 && n2%i==0){
        gcd = i;
    }
    find_gcd(n1, n2, i+1, gcd);
}
main(){
    find_gcd(24,36, 1, 0);
```

logic ⇒

n1=24
n2=36
i=1,
gcd=0;

if(1 > 12){ ↓ False

if(24%1==0 && 24%1==0) ↓ true
gcd=1;

find_gcd(24,36, 2, 1)

i=2
gcd=1
if(2 > 12) ↓ False
(24%2==0 && 36%2==0){ ↓ true
gcd=2;
Find_index_of(24,36, 3, 2)

i=3 ....this upto 12
gcd=2
if(3%2==0) ↓ False
(24%3==0) && (36%3==0) ↓ return false.
gcd=2;
Find_index_of(24,36, 4, 2)

i=4
gcd=2
if(4 > 12) ↓ False
(24%4==0) && (36%4==0) ↓ true
gcd=2
Find_index_of(24,36, 5, 2)

i=5
gcd=2
if(5 > 12) ↓ False
(24%5==0) && (36%5==0) ↓ return false.
gcd=2;
Find_index_of(24,36, 6, 2)

i=6
gcd=2
if(6 > 12) ↓ False
(24%6==0) && (36%6==0) ↓ true
gcd=2
Find_index_of(24,36, 7, 2)

i=7
gcd=2
if(7 > 12) ↓ False
(24%7==0) && (36%7==0) ↓ return false.
gcd=2;
Find_index_of(24,36, 8, 2)

i=8
gcd=2
if(8 > 12) ↓ False
(24%8==0) && (36%8==0) ↓ return false.
gcd=2;
Find_index_of(24,36, 9, 2)

i=9
gcd=2
if(9 > 12) ↓ False
(24%9==0) && (36%9==0) ↓ return false.
gcd=2;
Find_index_of(24,36, 10, 2)

i=10
gcd=2
if(10 > 12) ↓ False
(24%10==0) && (36%10==0) ↓ return false.
gcd=2;
Find_index_of(24,36, 11, 2)

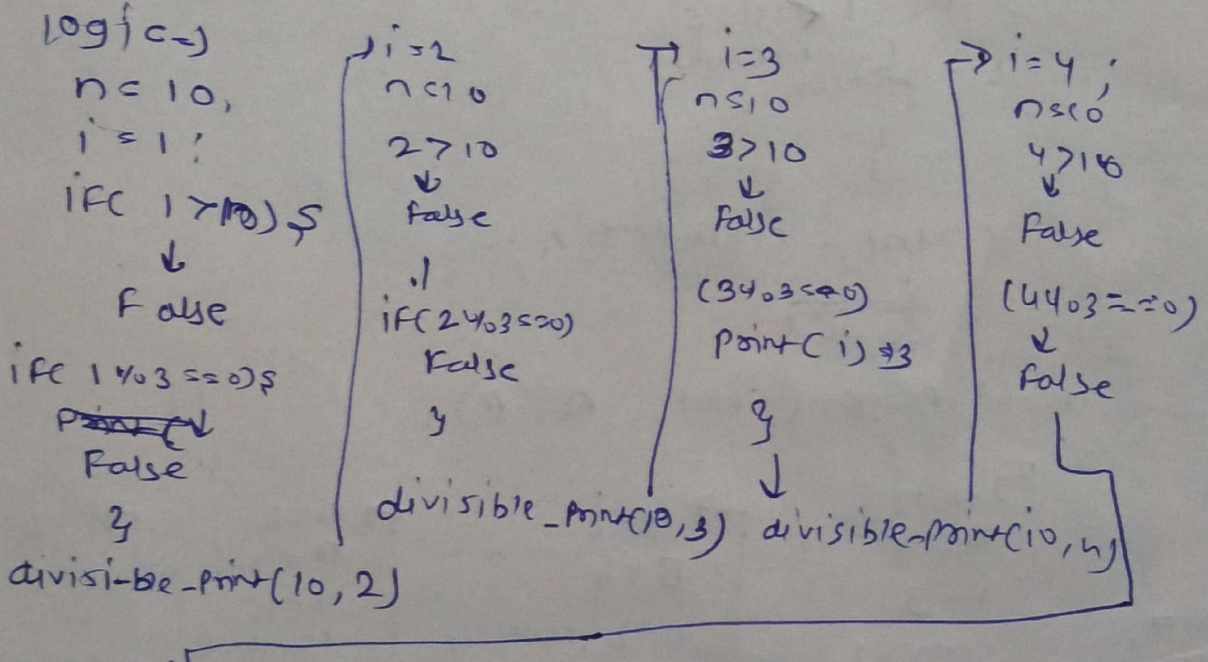
i=11
gcd=2
if(11 > 12) ↓ False
(24%11==0) && (36%11==0) ↓ return false.
gcd=2;
Find_index_of(24,36, 12, 2)

i=12
gcd=6
if(12 > 12) ↓ False
(24%12==0) && (36%12==0) ↓ true
gcd=12
Find_index_of(24,36, 13, 12)

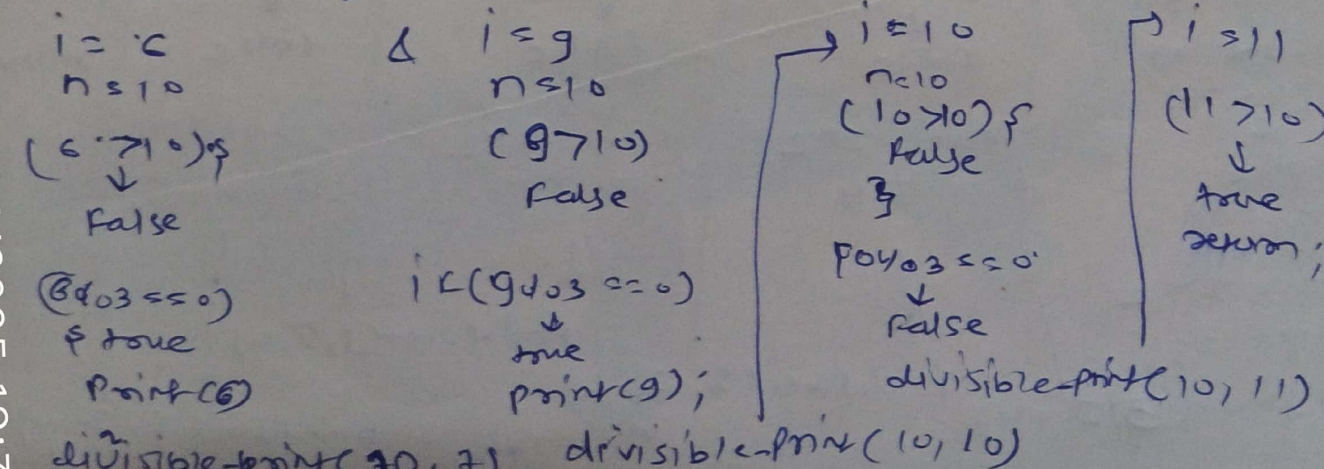
i=13
gcd=12
if(13 > 12) ↓ True
Print(12);
return;

[4] print all Numbers from 1 to N divisible by 3.

```
public static void divisible_Print( int n, int i){
    i=1; int count=0;
  if( i > n){
    return ; n
    }
  if( i%3 ==0){
     Print( i);
     }
divisible_print( n , i+1);
}.
```



similarly, 5, 6, 7 8 9



15] Find a power of Numbers using recursion.

```
Public static void Find_Power ( int A , int B) {
    int tempP_B = B;
    int count = 0;
    if ( B != 0) {
        return;
    }
```

```
Public static void Find_Power( int A, int B) {
    int i = B;
    int count = 1;
    if ( i == 0) {
        Print(count);
        return;
    }
    count = count * A;
    Find_Power( i-1, A, B);
```

logic ⇒

A = 2
B = 4
~~tempP = B = 4~~;
i = B = 4;
count = 1;
if (1 <= 0)
(4 <= 0)
↓
false
}
count = 1 * 2;
count = 2;
Find_Power(B12, 4)

A = 2
B = 4
i = 3
count = 2
if (3 <= 0)
↓
false
count = 2 * 2;
= 4
↓
Find_Power(2, 2, 4)

A = 2
B = 4
i = 2, count = 4
if (2 <= 0)
↓
false
count = 4 * 2
count = 8
↓
Find_Power(1, 2, 4)

A = 2
B = 4
i = 1,
count = 8
if (1 <= 0)
↓
false
count = 8 * 2;
count = 16
↓
Find_Power(0, 2, 4)

A = 2
B = 4
i = 0
count = 16
Print(16)
return;