Recursion: Recursion is the process of repeating items in a self-similar way. In programming languages, if a program allows you to call a function inside the same function, then it is called a recursive call of the function

Example- Print the sum of the series 1+2+…..+10 using recursion and without using loop

```
int Series(int);
int main(){
    int x = 10;
    printf("%d", Series(x));
    return 0;
}
int Series(int n){
    if(n == 1){
        return 1; // base case
    }
    else{
        return n + Series(n-1); // recursive case
    }
}
```

What happens after the first function call?

Series(10)
10 + Series(9)
10 + 9 + Series(8)
10 + 9 + 8 + Series(7)
…….. …… ……
10 + 9 + ……. + 3 + 2 + **Series(1)**
So the base case occurs here
10 + 9 + ……. + 3 + 2 + 1

Calculating factorial

Code (Non-Recursive Method)

```
#include <stdio.h>
int factorial(int x);
int main()
{
   int a,fact;
   printf("Enter a number:");
   scanf("%d",&a);
   fact=factorial(a);
   printf("Factorial value= %d", fact);

   return 0;
}
int factorial(int x)
{
   int f=1,i;
   for(i=x; i>=1; i--)
   {
      f=f*i;
   }
   return f;
}
```

Code Recursive Method

```
#include<stdio.h>
int factorial(int x);
int main()
{
   int a,fact;
   printf("Enter a number:");
   scanf("%d",&a);
   fact=factorial(a);
   printf("Factorial value= %d \n",fact);
   return 0;
}

int factorial(int x)
{
   if(x==1) //terminating or base case
      return 1;
   else
      return x*factorial(x-1);
}
```

1. Take input a number and then find the sum of the digits using recursion. [Hint: Use the remainder and division operators].

2. Write a C program to check if a number is prime number or not by recursion.

Dynamic Memory Allocation

In C, the exact size of array is unknown until compile time. Dynamic memory allocation allows your program to obtain more memory space while running, or to release it if it's not required.

Function	Use of Function
malloc()	Allocates requested size of bytes and returns a pointer first byte of allocated space **-malloc (number *sizeof(int));**
calloc()	Allocates space for an array elements, initializes to zero and then returns a pointer to memory **-calloc (number, sizeof(int));**
free()	deallocate the previously allocated space **-free (pointer_name);**
realloc()	Change the size of previously allocated space **-realloc (pointer_name, number * sizeof(int));**

Note: calloc () function is also like malloc () function. But calloc () initializes the allocated memory to zero. But, malloc() doesn't.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int *ptr, i , n1, n2;
    printf("Enter size of array: ");
    scanf("%d", &n1);

    ptr = (int*) malloc(n1 * sizeof(int));

    printf("Address of previously allocated memory: ");
    for(i = 0; i < n1; ++i)
         printf("%u\t",ptr + i);

    printf("\nEnter new size of array: ");
    scanf("%d", &n2);
    ptr = realloc(ptr, n2);
    for(i = 0; i < n2; ++i)
         printf("%u\t", ptr + i);
    return 0;
}
```

1. Find Smallest Element Using Dynamic Memory Allocation - calloc(). All elements should be taken as input.