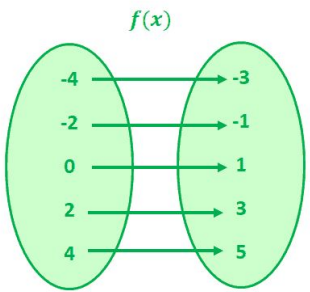
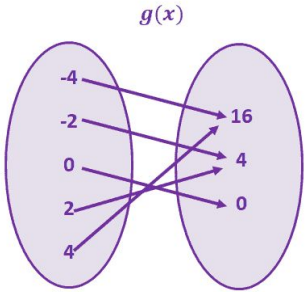




Functions

Week 10

Sumaiyah Zahid

Function Definition

A function is a black box which relates input to output.

INPUT

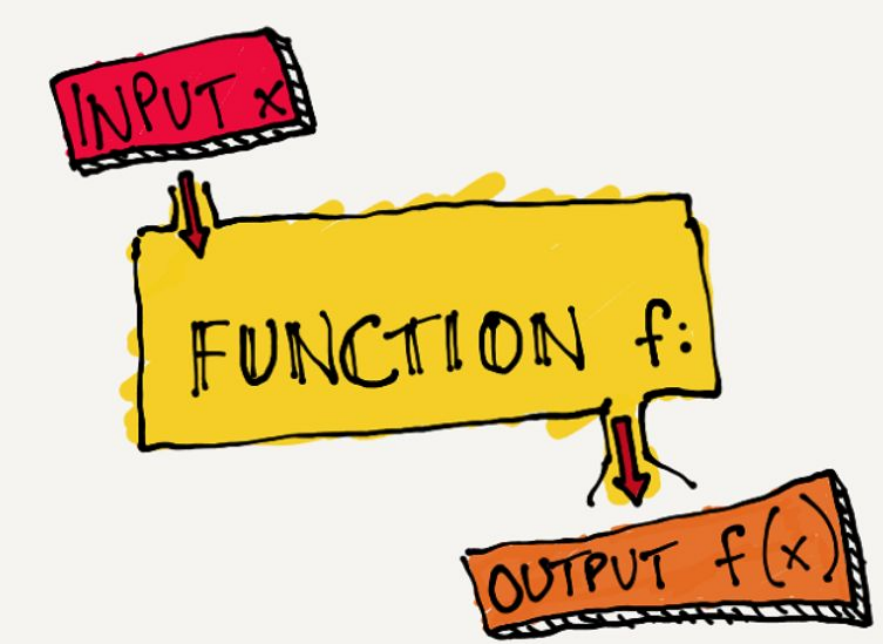
MYSTERY

OUTPUT

Outer World don’t know what happens inside.
Your relatives think it’s fun but in actual it isn’t.

Function

```
Output  Function_Name ( Input )
int main(void) //void = input
{
        return 0; // Output
}
Function name = main
Output datatype = int
```



Input / Arguments / Parameters

```
printf("Hello World");

scanf ("%d", &a);
```

printf and scanf are functions.

Inputs are separated by commas.

Inputs are also known as arguments or parameters.

Output / Return Values

```
a=printf("Hello World");

b=scanf ("%d", &a);

printf("%d %d", a,b);   // Try this
```

printf returns an integer value, which is the total number of printed characters.

scanf returns an integer value, which is the total number of inputs.

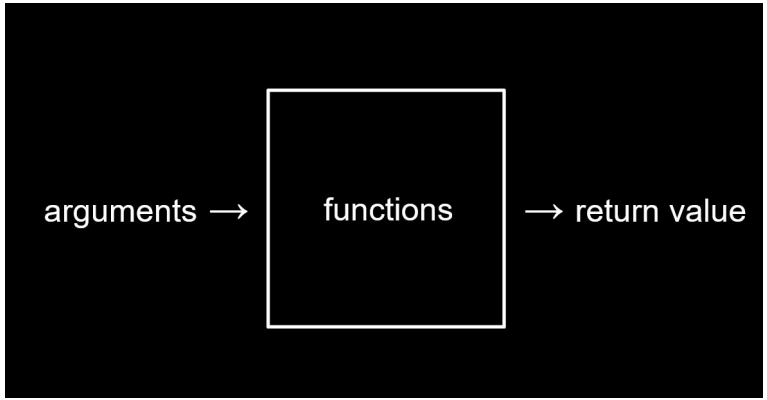
Function Definition

A block of code that performs a specific task.

They avoid duplicating codes.

Function help you break down a big project.

- Library Functions
- User - Defined Functions



User - Defined Functions

```
ReturnType  FunctionName (Arguments);
```

```
int add (int a, int b)
```

3 important things:

- Function declaration or prototype
- Function call
- Function definition

```
#include <stdio.h>

void functionName()
{
    ... .. ...
    ... .. ...
}

int main()
{
    ... .. ...
    ... .. ...

    functionName();

    ... .. ...
    ... .. ...
}
```

```c
#include <stdio.h>
void smile();     // Function Declaration or Prototype
int main()
{
   smile();       // Function Call
   return 0;
}

void smile()      // Function Definition
{
   printf("\nSmile, and the world smiles with you...");
}
```

```c
#include <stdio.h>
void smile()// Function Declaration & Definition
{
   printf("\nSmile, and the world smiles with you...");
}
int main()
{
    smile();        // Function Call
    return 0;
}
```

Practice Question

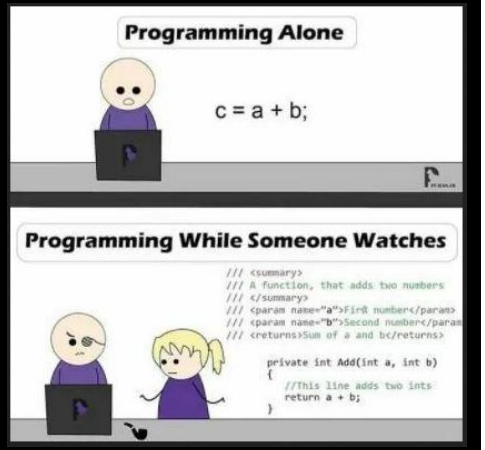
Make a function in C which prints following pattern.

```
****

****

****

****
```

```c
#include <stdio.h>
void star();    // Function Declaration or Prototype
int main()
{
   star();        // Function Call
   return 0;
}

void star()    // Function Definition
{
   int i;
   for(i=0; i<4; i++)
      printf("\n****");
}
```

```c
#include <stdio.h>
void add(int a, int b); // Function Prototype
int main()
{
   add(5,4);          // Function Call
   return 0;
}

void add(int a, int b)     // Function Definition
{
   printf("%d", a+b);
}
```

```c
#include <stdio.h>
int add(int a, int b);
int main()
{
   int sum;
   sum=add(5,4);
   printf("%d", sum);   // you can’t access a and b here
   return 0;
}

int add(int a, int b)
{
    int c=a+b;          // you can’t access sum here
    return c;
}
```



Why Functions?

To hide irrelevant detail at the main() level, so the the program's primary purpose is clearer.

To divide a complex problem into a series of simple problems

To make subsequent modification of the program easier.

To reduce the errors that inevitably come with a single large complex program

Practice Question

Make a function in C which takes one int argument and check whether it’s an even or odd.

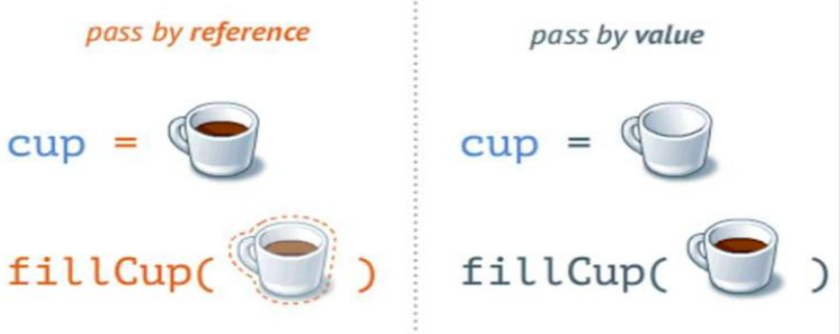
```c
#include <stdio.h>
int even_odd(int x);
int main()
{
    int num=6;
    even_odd(num);
    return 0;
}
int even_odd(int x)
{
    if (x%2==0)
        printf("It's an even number!");
    else
        printf("It's an odd number!");
}
```

Practice Question

Make a function in C which takes int as a parameter and return it’s cube value.

```c
#include <stdio.h>
int cube(int a);
int main()
{
   int num=4, result;
   result=cube(num);
   printf("%d", result);
   return 0;
}

int cube(int a)
{
    return a*a*a;
}
```

Calling Functions



Call by value

- Copy of argument passed to function
- Changes in function do not effect original
- Use when function does not need to modify argument
- Avoids accidental changes

All the examples mentioned before are using call by value.

Call by reference (We will study later with Pointers)

- Passes original argument
- Changes in function effect original
- Only used with trusted functions

Passing Array Elements

Passing array elements to a function is similar to passing variables to a function.

```
void display(int age1, int age2) {
}
int main() {
  int ageArray[] = {2, 8, 4, 12};
  display(ageArray[1], ageArray[2]);
  return 0;
}
```

Passing Array

1D array:

```
void myFunction(int arr[10]);
void myFunction(int arr[]);
```

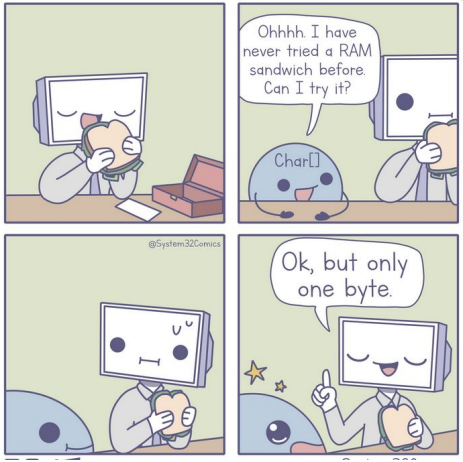
2D array:

```
void myFunction(int arr[10][10]);
void myFunction(int arr[][10]);
```

```c
#include <stdio.h>
float calculate_sum(float arr[])   ;
int main()
{
   float array[5]={1.5, 2.3,4.5,6.7,1.3};

   printf("%f", calculate_sum(array));
   return 0;
}

float calculate_sum(float arr[])
{
     float sum=0; int i;
     for(i=0; i<5; i++)
        sum=sum+arr[i];
     return sum;
}
```

```c
#include <stdio.h>
float calculate_sum(float arr[], int size);
int main()
{
   float array[5]={1.5, 2.3,4.5,6.7,1.3};

   printf("%f", calculate_sum(array,5));
   return 0;
}

float calculate_sum(float arr[], int size)
{
     float sum=0; int i;
     for(i=0; i<size; i++)
        sum=sum+arr[i];
     return sum;
}
```

```c
#include <stdio.h>
float calculate_sum(float arr[2][5])   ;
int main()
{
   float array[2][5]={{1.5, 2.3,4.5,6.7,1.3},
                      {10.3,6.5,3.9,6.7,8.3}};

   printf("%f", calculate_sum(array));
   return 0;
}

float calculate_sum(float arr[2][5])
{
     float sum=0; int i,j;
     for(i=0; i<2; i++)
        for (j=0; j<5; j++)
            sum=sum+arr[i][j];
     return sum;
}
```


Ohhhh. I have never tried a RAM sandwich before. Can I try it?
Char[]
@System32Comics
Ok, but only one byte.

STRINGS

String \ Char array

```
char name[]={‘F’,’A’,’S’,’T’,’ ‘ ,’N’, ‘U’, ‘\0’};
char name[]="FAST NU";
```

\0 = Null character or String Terminator

```
for(i=0; i<10; i++)
   { printf("%c",name[i]);}
```

String

```
#include <stdio.h>
int main()
{
    char name[20];
    scanf("%s", name); // No need of &name
    printf("Your name is %s", name);
    return 0; // Array name alone works as a base address
}
```

Gets and Puts

```
#include <stdio.h>
int main()
{
    char name[20];
    puts("Enter your name");
    gets(name);
    puts(name);
    return 0;
}
```

2D Strings

J	a	v	a	\0					
P	y	t	h	o	n	\0			
C	+	+	\0						
H	T	M	L	\0					
S	Q	L	\0						

```
char language[5][10] =
{
  {'J','a','v','a','\0'},   {'P','y','t','h','o','n','\0'},
  {'C','+','+','\0'},  {'H','T','M','L','\0'},
  {'S','Q','L','\0'} };
```

```
char language[5][10] = {"Java", "Python", "C++", "HTML",
"SQL"};
```

2D Strings

```
// it is valid
char language[ ][10] = {"Java", "Python", "C++", "HTML", "SQL"};

// invalid
char language[ ][ ] = {"Java", "Python", "C++", "HTML", "SQL"};
// invalid
char language[5][ ] = {"Java", "Python", "C++", "HTML", "SQL"};
```

```c
#include <stdio.h>
int main()
{   int i;
    char language[5][10] = {"Java",
"Python", "C++", "HTML", "SQL"};

    for(i=0; i<5; i++)
        printf("%s\n", language[i]);
    return 0;
}
```

```c
#include <stdio.h>
int main()
{   int i;
    char name[5][10];
   for(i=0; i<5; i++)
        scanf("%s", name[i]);

    for(i=0; i<5; i++)
        printf("%s\n", name[i]);
    return 0;
}
```

HOME ASSIGNMENT

Write a program to change the case of all the alphabets in an array of strings.

Write a program that counts the no. of upper and lower case letters in an array of strings

String.h ------> strlen

```
size_t strlen(const char *str);
```

Computes the length of the string str up to but not including the terminating null character.

Returns the number of characters in the string.

String.h ------> strcmp

```
int strcmp(const char *str1, const char *str2)
```

It compares the two strings and returns an integer value.

- If Return value < 0 then it indicates str1 is less than str2.
- If Return value > 0 then it indicates str2 is less than str1.
- If Return value = 0 then it indicates str1 is equal to str2.

String.h ------> strncmp

```
int strncmp(const char *str1, const char *str2, size_t n)
```

It compares both the string till n characters or in other words it compares first n characters of both the strings.

```c
#include <stdio.h>
#include<string.h>
int main()
{
    char name1[20]="FAST";
    char name2[20]= "NUCES";
    printf("Length of string is %d and %d \n",
    strlen(name1), strlen(name2));
    int i=strcmp(name1, "FAST");
    int j=strcmp(name1, name2);
    printf("Comparison result is %d %d", i,j);
    return 0;
}
```

String.h ------> strcat

```
char *strcat(char *str1, char *str2)
```

It concatenates two strings and returns the combined string.

String.h ------> strncat

```
char *strncat(char *str1, char *str2, int n)
```

It concatenates n characters of str2 to string str1.

String.h ------> strcpy

```
char *strcpy(char *str1, char *str2)
```

It copies the string str2 into string str1, including the end character (terminator char ‘\0’).

String.h ------> strncpy

```
char *strncpy(char *str1, char *str2, int n)
```

It copies the n characters of str2 into string str1.

Try These

```
strcat(str1, str2)

strncat(str1,str2, 5)

strcpy(str1,str2)

strncpy(str1,str2,5)
```

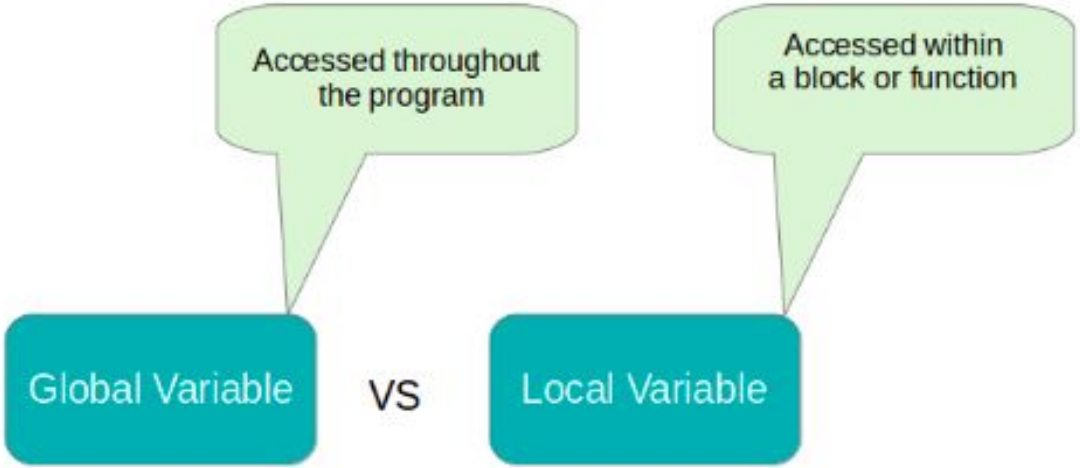
https://fresh2refresh.com/c-programming/c-strings/

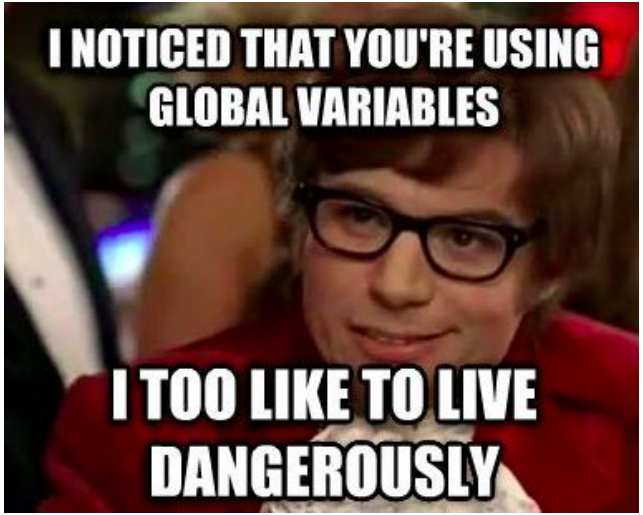
String functions	Description
strcat ( )	Concatenates str2 at the end of str1
strncat ( )	Appends a portion of string to another
strcpy ( )	Copies str2 into str1
strncpy ( )	Copies given number of characters of one string to another
strlen ( )	Gives the length of str1
strcmp ( )	Returns 0 if str1 is same as str2. Returns <0 if strl < str2. Returns >0 if str1 > str2

https://fresh2refresh.com/c-programming/c-strings/

strchr ( )	Returns pointer to first occurrence of char in str1
strrchr ( )	last occurrence of given character in a string is found
strstr ( )	Returns pointer to first occurrence of str2 in str1
strrstr ( )	Returns pointer to last occurrence of str2 in str1
strdup ( )	Duplicates the string
strlwr ( )	Converts string to lowercase
strupr ( )	Converts string to uppercase
strrev ( )	Reverses the given string
strset ( )	Sets all character in a string to given character
strnset ( )	It sets the portion of characters in a string to given character
strtok ( )	Tokenizing given string using delimiter

Local Vs Global Variables





```c
#include <stdio.h>
int global=20;
void smile();
int main()
{  int local=10;
   printf("Local Value is %d\n", local);
   printf("Global Value is %d\n", global);
   smile();
   printf("Global Value is %d\n", global);
   return 0;
}
void smile()
{  global+=20;
   printf("\nSmile, and the world smiles with you...");}
```

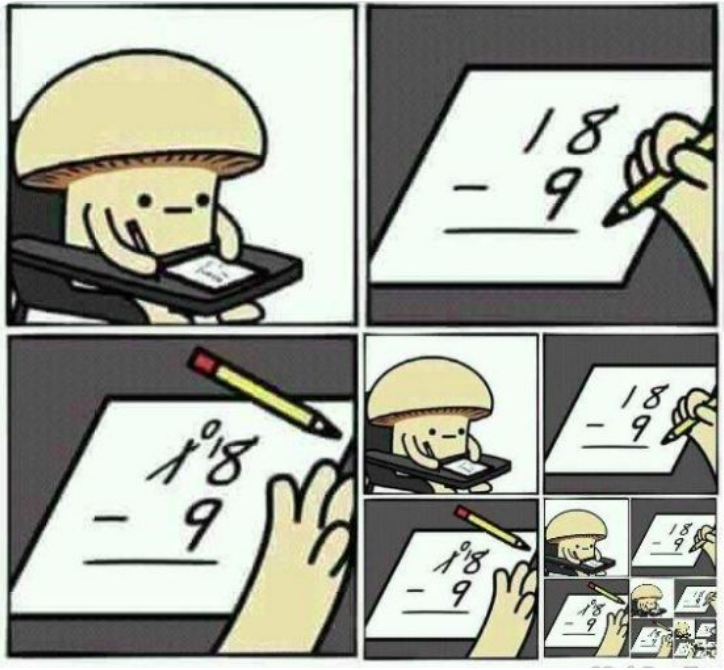
Nested Functions Calling

We can call another user-defined function inside any user-defined function.

Make a function add which takes two argument and decides whether it’s sum is even or odd.

```c
#include <stdio.h>
int add(int a, int b);
void even_odd(int x);
int main()
{
    int sum;
    sum=add(5,4);
    printf("%d", sum);
    return 0;
}
```

```
int add(int a, int b)
{
    int c=a+b;
    even_odd(c);
    return c;
}
void even_odd(int x)
{
    if (x%2==0)
        printf("It's an even number!");
    else
        printf("It's an odd number!");
}
```



RECURSION

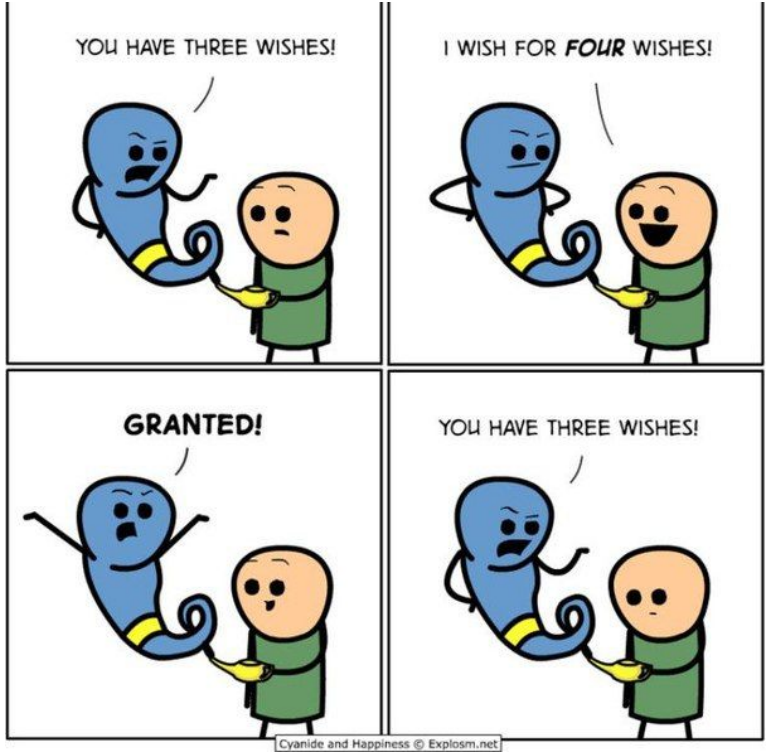
See recursion

RECURSION

A function calling itself.

```
void recurse()
{
    ... .. ...
    recurse();
    ... .. ...
}

int main()
{
    ... .. ...
    recurse();
    ... .. ...
}
```

recursive call



RECURSION

fact(n) : n!

fact(1)=1

fact(2)=2*1

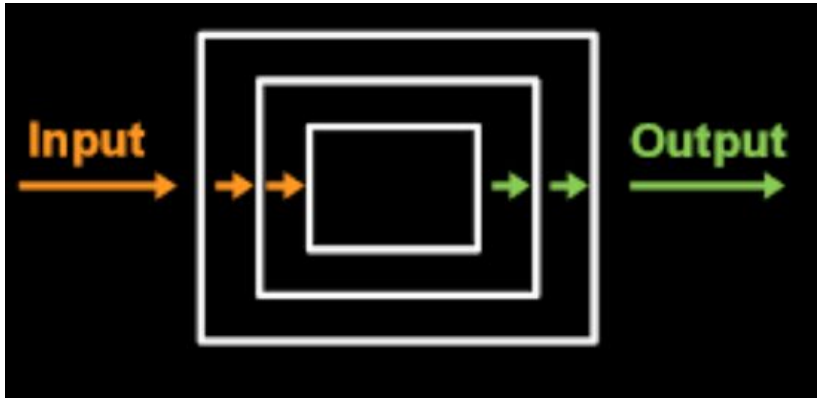
fact(3)=3*2*1

fact(4)=4*3*2*1

fact(5)=5*4*3*2*1

RECURSION

```
fact(n) : n!
fact(1)=1
fact(2)=2*fact(1)
fact(3)=3*fact(2)
fact(4)=4*fact(3)
fact(5)=5*fact(4)
```



RECURSION

```
fact(n) : n * fact(n-1)
```

Recursive function have 2 cases:

- Base Case
- Recursive Case

RECURSION

fact(n) : n!

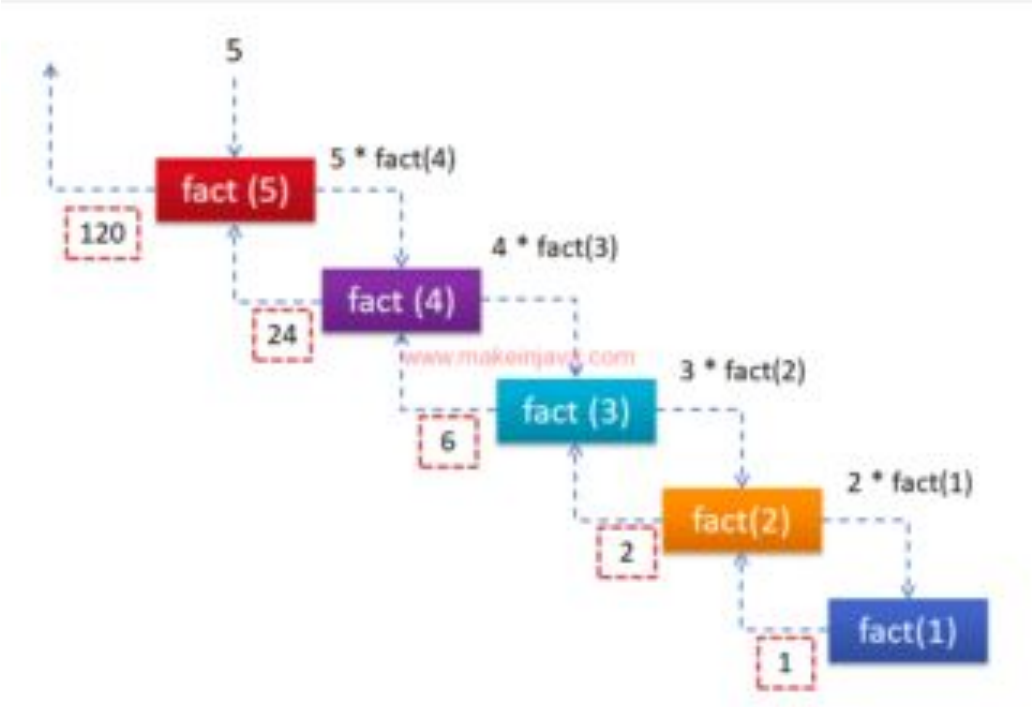
fact(1)=1

fact(2)=2*fact(1)

fact(3)=3*fact(2)

fact(4)=4*fact(3)

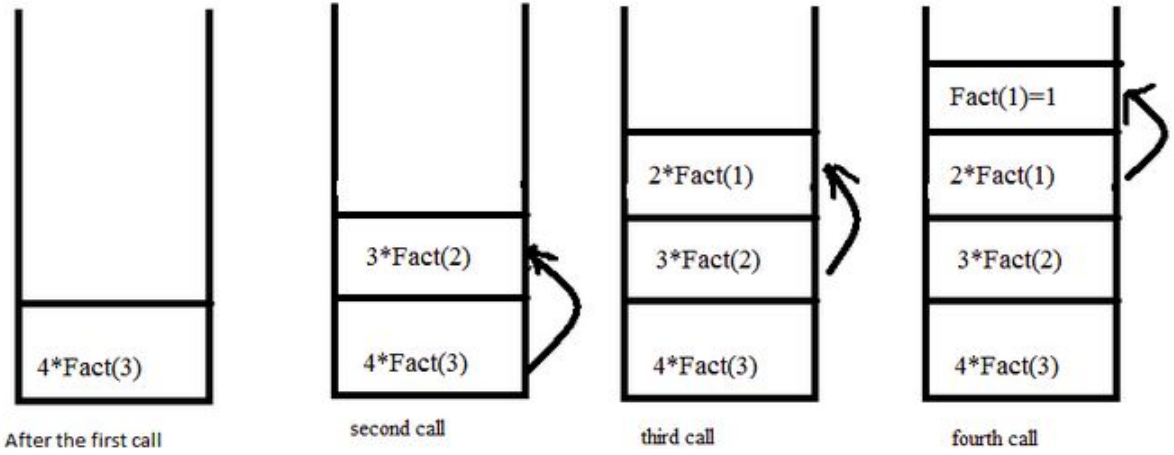
fact(5)=5*fact(4)



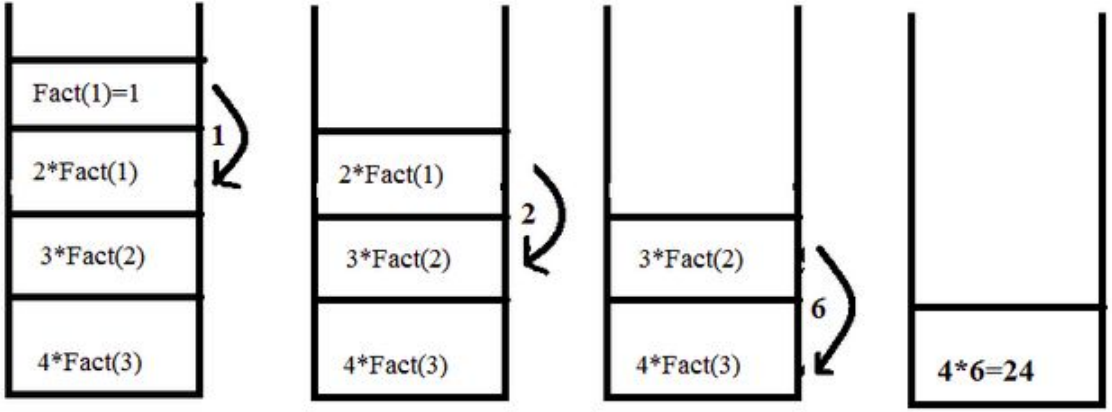
Factorial Function

```
int factorial( int n)
{
    if (n==1)           // Base case
        return 1;
    else                // Recursive Case
        return n * factorial(n-1);
}
```

When function call happens previous variables gets stored in stack



Returning values from base case to caller function



Practice Question

Write a program in C to calculate the sum of numbers from 1 to n using recursion.

Sum Function

```
int sum( int n)
{
    if (n==1)           // Base case
        return 1;
    else                // Recursive Case
        return n + sum(n-1);
}
```

Practice Question

Write a program in C to print Fibonacci sequence to n terms using recursion.

Fibonacci Function

```
int fibonacci( int n)
{
   if (n==0)                // 1st Base case
         return 0;
   else if (n==1)           // 2nd Base case
         return 1;
   else                     // Recursive Case
         return fibonacci(n-2) + fibonacci(n-1);
}
```

Home Assignment

Write a program in C to print first 50 natural numbers using recursion.

Write a program in C to print the array elements using recursion.

Write a program in C to count the digits of a given number using recursion