FUNCTION

DATE : __/__/__
PAGE : ____

* Function :—

→ A function is a program. It is also a group of statement that grouped together to perform a task.

→ A function always return value except void data type.

Syntax :—

```
return-type function-name (parameter list)
{
    statement;
}
```

* Type of function :—

(I) Pre-defined function
(II) User-defined function

(I) Pre-defined function / Library function / Inbuilt function

→ The function which meaning is already defined in C language library is known as pre-defined function or library function.

Ex :—
getch (), clrscr (), strlen (), strcat (), strcmp (), strcpy (), strupr (), gets (), puts () etc.

(II) User-defined function :-

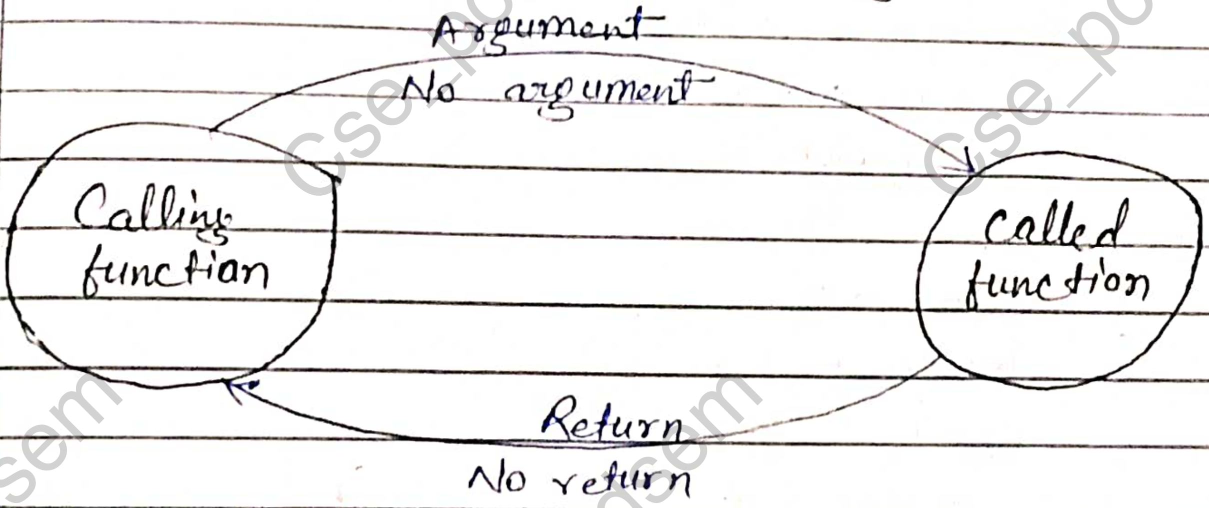
↳ User-defined function is that function which is defined by the user.

Ex :-

add(), sub(), mul(), div() etc.

There are four types of user defined function :-

(i) No return with no argument passing
(ii) No return with argument passing
(iii) Return with no argument passing
(iv) Return with argument passing



Note :- Main function is always calling function & sub function may be calling function or may be called function.

DATE: __/__/__
PAGE:

(i) No return with no argument passing :-

→ Whenever calling function doesn't pass the arguments to the called function & also called function doesn't return the any values to the calling function, is known as "no return with no argument passing".

Ex:-

Write a program to find the addition of two number using function.

```
#include<stdio.h>
#include <conio.h>
void add()
{
~~clrscr();~~
int a,b,c;
printf("\n enter the value of a & b");
scanf("%d %d",&a,&b);
c=a+b;
printf("\n sum=%d",c);
}
void main()
{
   clrscr();
   add();
   getch();
}
```

DATE: ___/___/___
PAGE: ___

(ii) No return with argument passing.

→ In this type of function, calling function will passes the arguments to the called function but called function will not return any value to the calling function.

Ex :—

WAP to find the addition of two number using function.

```
#include <stdio.h>
#include <conio.h>
void add(int x, int y)      —— formal argument
{
    int z;
    z = x + y;
    printf("\n sum = %d", z);
}

void main()
{
    int a, b;
    clrscr();
    printf("\n enter the two values");
    scanf("%d %d", &a, &b);
    add (a,b);  ——→ Actual argument
    getch();
}
```

DATE : __/__/__
PAGE : ____

(iii) Return with no argument passing :-
→ In this type of function, calling function will not passes any argument to the called function but called function will return a value to the calling function.

Ex:-

WAP to find the addition of two no. using function.

```
#include <stdio.h>
#include <conio.h>
~~void main()~~ int add()
{
   int a,b,c;
   printf("\n enter the value of a & b");
   scanf("%d %d", &a, &b);
   c = a+b;
   return c;
 }
 void main()
 {
   int s;
   clrscr();
   s = add();
   printf("\n sum= %d", s)
  getch();
}
```

DATE : __/__/__
PAGE :

(iv) Return with argument passing :-

→ In this type of function, calling function passes the arguments to the called function and called function will also return the value to the calling function.

Ex :-

WAP to find the addition of two no. using function.

```
#include <stdio.h>
#include <conio.h>
int add (int a, int b)
{
  int c;
  c = a + b;
  return c;
 }
 int main()
 {
  int a, b, c;
  printf ("\n enter the value of a & b");
  scanf ("%d %d", &a, &b);
  c = add (a, b);
  printf ("\n sum = %d", c);
  getch ();
  return 0;
 }
```

DATE : __/__/__
PAGE :

* Prototype function :-

It is a type of function, in which the name of sub function is define below the header file and above the main function and the whole sub program is defined/written below the main function is known as prototype function.

Ex:-

WAP to find the addition of two no's.

```
# include <stdio.h>
# include <conio.h>
void add ();
/*prototype function*/
void main()
{
 clrscr();
 add();
 getch();
}

void add()
{
 int a, b, c
 printf("\n enter the value of a & b");
 scanf("%d %d", &a, &b);
 c=a+b ;
 printf("\n sum=%d", c);
}
```

8. WAP to calculate the factorial of any given no. & also check ~~the~~ even or odd no. using prototype function.

#

#

DATE : ___/___/___
PAGE : ______

Recursion :—

Recursion is a type of function whenever a function call itself is known as recursion.
Or,

we know that a function calls another function or sub function whenever a function called if itself is known as recursion.

Ex:—

WAP to find a factorial of any given no. using recursion.

```
#include <stdio.h>
#include <conio.h>
 int fact (int n);
 void main ()
 {
   int n, f;
```

DATE : __/__/__
PAGE : ____

```
clrscr();
printf("\n enter the value of n");
scanf("%d", &n);
f = fact (n);
printf("\n factorial = %d", f);
getch();
}

int fact (int n)
{
int v = 1;
if (n == 0)
  return v;
else
  v = n x fact (n-1);
  return v;
}
```

Dry run:-

Enter the value of n.

5

5 = = 0 ✗

v = 5 x fact (4)

4 = = 0 ✗

v = 5 x 4 x fact (3)

3 = = 0 ✗

v = 5 x 4 x 3 x fact (2)

2 = = 0 ✗

V = 5 × 4 × 3 × 2 × fact (1)
1 == 0 ×
V = 5 × 4 × 3 × 2 × 1 × fact (0)
0 == 0 ✓
V = 120 × 1 = 120
factorial = 120

* Storage class :—
A storage class defines the scope (visibility) and life times of variables and/or function within a C program is called storage class.
They proceed the type that they modify.
There are four types of storage class :—
i) Auto (Automatic Storage Class)
ii) Register (Register Storage Class)
iii) Static (Static Storage Class)
iv) Extern (Extern Storage Class)

i) Automatic Storage Class :—
Automatic storage class is the default storage class for all local variables.
Ex :—

```
{
    int x;
    auto int x;
}
```

In this example two variables with-in the same storage class. Auto can only be used within function i.e. local variable.

DATE : __/__/__
PAGE :

ii) Register Storage Class :-

The register storage class is used to define local variables that should be stored in a register instead of RAM. i.e. means the variable has a maximum size equal to the register size & can't have the uniary AND operator applied to it.

Ex :-

```
{
    register int x;
}
```

iii) Static Storage Class :-

The static storage class instructs the compiler to keep a local variable in existance during the life time of the program instead of creating and destroying it each time. It comes into and goes out of scope, therefore making local variable. Static allows them to maintain their values between function calls.

The static modify may also applied to the global variable.

In C language, when static is used on global variable, it cause only one copy of that member to share by all the

object of its class.
Ex:—

```
#
void abc();
static int count = 5;  /* Global variable */
void main()
{
  while (count--)
  {
    abc ();
  }
  return 0;
}
void abc ()
{
  static int i = 5;  /* local variable. */
  i++
  printf ("\n i is %d and count is %d", i, count)
}
```

iv) Extern Storage Class:—

The extern storage class is used to give a refrence of a global variable i.e. visible to all the program files.

When we use 'extern', the variable can't be initialized however, it points the variable name at a storage location that has been previously defined.