a) Functions without arguments and without return type.

-check whether the year is Leap year

```
#include <stdio.h>
void check (void);
  main()
 {
 check();




}
void check ()
{
int x ,y;
printf("enter:");
scanf("%d",&x);
y=x%4;

if(y==0)
printf("Leap year");
else
printf(" Not leap year");

}
```

Output:-

```
Enter :2012
```

Leap year

-count number of digits in a number.

```
#include <stdio.h>
void digit (void);
  main()
 {
 digit();
}
void digit ()
{
int i,a,b,c,cnt=0;
printf("enter");
scanf("%d",&a);
for(i=a; i != 0;)
{
b=i%10;
c=i/10;
cnt=cnt+1;
i=c;
}
printf("Digit=%d",cnt);
}
```

Output:-

```
Enter 123
Digit=3
```

b) Functions without arguments and with return type

--check Armstrong number or not.

```
#include <stdio.h>
 #include <math.h>
int armstrongNumberFinder(void);
int main()
{
 int  flag;

 flag = armstrongNumberFinder();
if (flag == 1)
printf("%d is an Armstrong number.");
 else printf("%d is not an Armstrong number.");
 return 0;
}
int armstrongNumberFinder(void)
 {
int num;
printf("entee");
scanf("%d",&num);
 int original, rem, sum = 0, n = 0, flag;
 original = num;
 while(original != 0)
{
original /= 10; ++n;

 }
original = num;
while(original != 0)
 {
```

```
 rem = original%10; sum += pow(rem, n); original /= 10;
 }
if(sum == num) flag = 1;
else flag = 0;
return flag;
}
```

c) Functions with arguments and without return type

-check prime number or not.

```
#include <stdio.h>
 void prime(int);
int main()
{
int a;
printf("ent");
scanf("%d",&a);
prime(a);



}
void prime (int x)
{
int i,flag=0;
for(i=2;i<=x/2;i++)
{
if(x%i==0)
{
flag=1;
break;
}
}

if(x==1)
printf("1 is a not a prime nor composite");

else if(flag==0)
printf("prime");

else
printf("not prime number");

}
```

Output:-

```
ent7
prime
```

<u>d) with Functions arguments and with return type.</u>

-**count number of digits in a number**

```
#include <stdio.h>
int digit (int);
  main()
 {
 int x,y;
printf("enter");
scanf("%d",&x);
y=digit(x);
printf("digit=%d",y);
}
int digit (int a)
{
int i,b,c,cnt=0;

for(i=a; i != 0;)
{
b=i%10;
c=i/10;
cnt=cnt+1;
i=c;

}

return cnt;
}
```

Output:-

```
enter6777
digit=4
[Process completed - press Enter]
```

-calculate factorial of a number.

```
#include <stdio.h>
int fact (int);
  main()
 {
 int x,y;
printf("enter");
scanf("%d",&x);
y=fact(x);
printf("Factorial=%d",y);


}
int fact (int a)
{
int i,b,c;
double mul=1;


for(i=1; i<=a;i++)
{
mul=mul*i;
}


return mul;
}
```

Output:-

```
enter6
Factorial=720
[Process completed - press Enter]
```

<u>g) Recursive Functions</u>

<u>-to print even or odd numbers in given range</u>

```
#include <stdio.h>
void evod (int,int);
  void main()
 {
 int x,y;
printf(" from ");
scanf("%d",&x);

printf("To ");

scanf("%d",&y);

evod(x,y);


}

void evod (int x,int y)

{

int i,b,c;

if(x>y)


return ;

printf("%d ,",x);


evod(x+2,y);


}
```

Output:-

```
from 1

To 20

1 ,3 ,5 ,7 ,9 ,11 ,13 ,15 ,17 ,19 ,
```

h) Passing 1D Array in Functions

- Reverse the elements of an array

```
#include <stdio.h>

void rev (int[]);


 void main()
```

```
{
int a,i;


int x[5]={1,2,3,4,5};

rev(x);

}
void rev (int p[])
{
int i;
for(i=4;i>=0;i--)
{
printf("\n%d",p[i]);

}


}
```