$S = 1 + 2^2 + 3^3 + 4^4 + 5^5 + 6^6 \ldots n^n$ Series.

Passing Arguments to function →

(I) Pass by value →

```
void main()
{
int x, y;
void swap(int, int);
x=10;
y=20;
swap(x,y);
cout<< x <<endl;
cout<<y<<endl;
getch();
void swap(int a, int b)
{
  int t;
  t=a;
  a=b;
  b=t;
}
```

10 → a, 20 → b

t = 10;
a = 20; ⇒ x = 20
b = 10; ⇒ y = 10;

No change

O/P
x = 10
y = 20

value of x & y are copied into a and b but the change in a & b does not affect into x & y.

(i) value of variable is passed

(ii) Both actual & formal parameter are stored at diff location i.e called fn can't access actual parameter.

(II) Pass by reference.

```
void swap (int &, int &)

swap( x,y);

void swap (int &a, int &b)
{
  int t;
  t=a;
  a=b;
  b=t;
}
```

Reference variable

O/P
x = 20
y = 10

a & b are reference name for x & y therfore they refer to same region. i.e change in a & b reflected the change in x & y;

(i) reference of original variable is passed

(ii) It works on original variable i.e actual variable rather than formal variable.

(iii) It access the actual parameter in called fn.

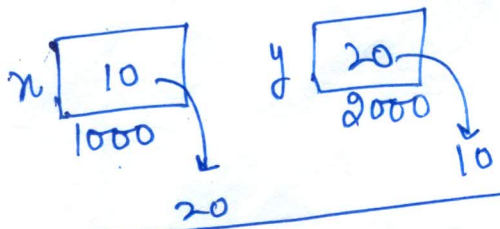
(III) Pass by address or Pointer

```
void swap(int *, int *)
int x = 10;
int y = 20;
swap(&x, &y) ⇒ address
```

```
void swap(int *a, int *b)
{
int t;
t = *a;
*a = *b;
*b = t
}
```

	*a, a = &x
	i.e *a(*1000)
	t=10;
	*a=20;
	*b=10;

O/P x = 20
y = 10



(IV) Return by Reference

```
int & min(int &a, int &b)
{
if (a<b)
return(a);
else
return(b);
}
```

Return type ⟹ Reference not value

min(&a, &b) = 10

Inline function ⇒

```
Inline int square(int a)
{ return(a*a);
}
void main()
{ int b;
cout << "Enter the No";
cin >> b;
cout << square(b);
getch();
}
```

(I) No prototype declare
(II) No complete fn is written before main.
(III) It is preferable when inline fn is small
(IV) It replace function call with function code but control not jump to function.

Default argument ⇒ (1) only trailing arguments can have default have

```
float add (int a, int b=10, int c=20)        ✓
float add (int a=10, int b=20, int c=20)     ✓
int add(int a=10, int b, int c)              x
int add (int a=10, int b, int c=10)          x
   add(x); add( );
```

Recursion ⇒ function call itself.

```
void main()
{
  f1();
}
void f()
{
 void f(); & local fn prototype
  f1();
  ==
}
```

factorial

```
{int fact (int);
cout<< "no";
cin>>n
int f= fact(n);
cout<<f;
getch();
}
int fact (int a)
{
int fact(int);
int value = a * fact(a-1);
return(value);
}
```

3! = 3 × fact(2) = 6
fact(2) = 2 × fact(1)
fact(1) = 1 (base case)

Function overloading

virtual function & friend function

const Argument

```
const x=10
int add(const b);
```