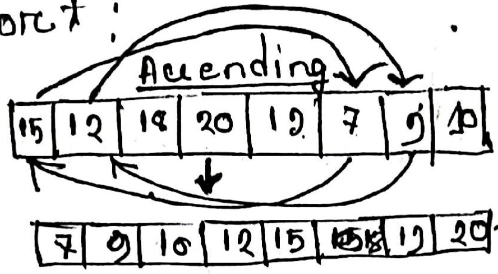
* Selection sort:



→ [যেটার ক্রমে থাকি সবার প্রথমে sort করার লাগবে না]

→ Sorted in ascending order

Step: i) Swap

ii) minimum element and its index.

iii) Selection Sort implementation.

* Swap two element:

a	b	temp
7	5	5

a = 5
b = 7

```
temp = a
a = b
b = temp
```

Arithmatic style

```
a = a+b
b = a-b
a = a-b
```

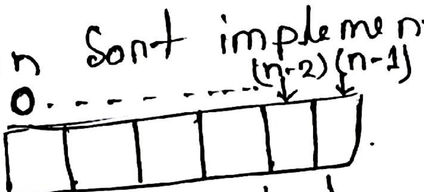
* minimum:

একটার একটার করে check করব যে minimum হবে সেটার index নিবো।

```
min = arr[0]
min_index = 0
for (int i=1; i<n; i++){
  if(arr[i] <min)
  {
    min = arr[i];
    min_index=i;
  }
```

(Time complexity → n) **

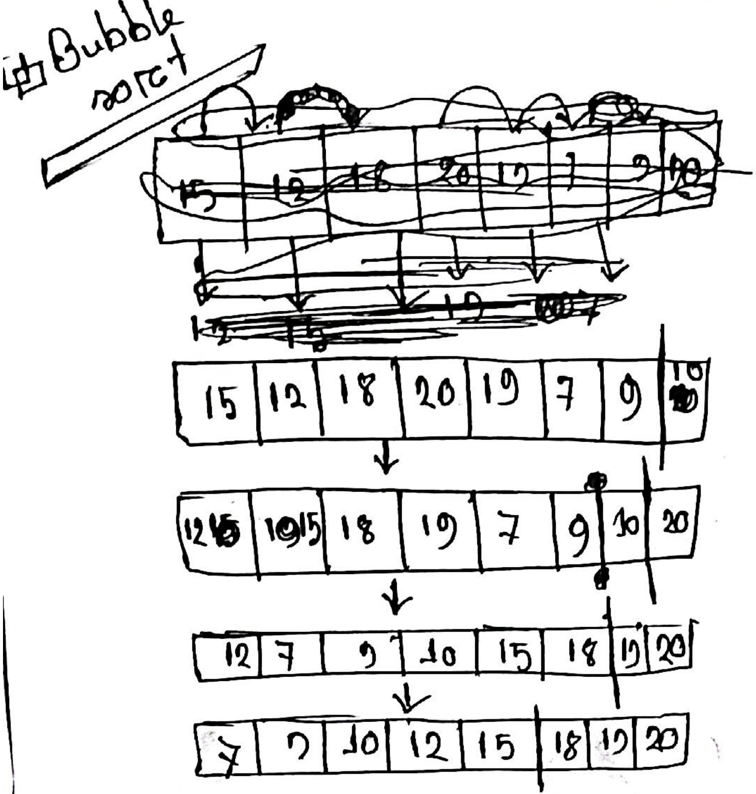
* Selection Sort implement:




```
for(int i=0; i<=n-2 ; i++)
{ min = arr[i];
  min_index = i;
  for(int j=i+1 ; j<n; j++){
  ~~min = arr~~
    if(arr[j] <min)
    {
      min= arr[j]
      min_index=j;
    }
  int temp = arr[i];
  arr[i] = arr [min index]
  arr [min_index] = temp
```

2

Bubble sort



Left   Right

arr[Left] > arr[right]
{ Swap arr[Left] &
  arr[right]
}

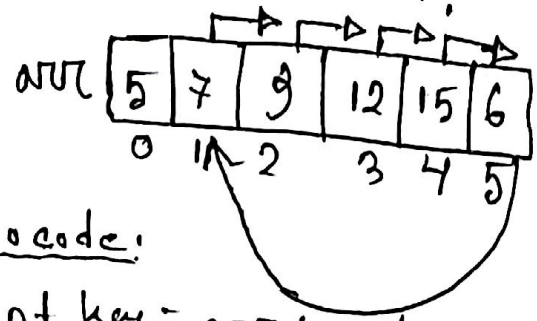
* [সবচেয়ে বড় element 1st iteration এর পর শেষে চলে যাবে এবং সেখানেই থাকবে]

Step: i) boundary = n-2 to 0
boundary -= 1;
→ Left = 0 to boundary

Sudo code:

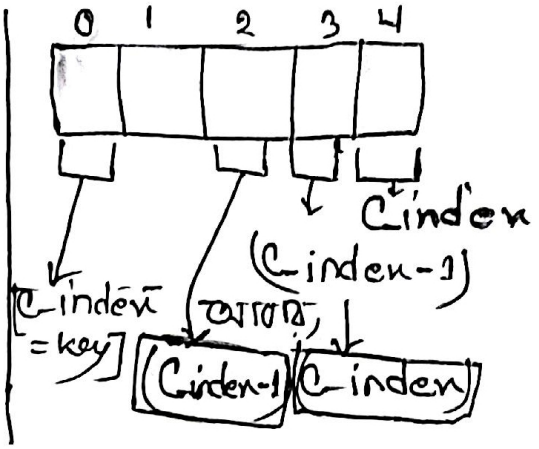
```
for (boundary = n-2 ; boundary >= 0 ; boundary--)
{  for (Left = 0; Left <= boundary ; Left++)
   {  if (arr[Left] > arr[Left+1])
      {  temep = arr[Left]
         arr[Left] = arr[Left+1]
         arr[Left+1] = temp
      }
   }
```

☐ insertation sort: ——▷ [assumption in only learn one in unsorted]



*Sudo code:

```
int key = arr(n-1);
int c_inden = n-1;
while(c_inden > 0 && arr[c_inden-1] > key){
    arr[c_inden] = arr[c_inden-1];
    c_inden -= 1;
}
arr[c_inden] = key;
```




```
for(int i=1; i<n; i++){
int key = arr[i];
int c_inden = i;
while(c_inden>0 && arr[c_inden-1] > key){
    arr[c_inden] = arr[c_inden-1];
    c_inden -= 1;
}
arr[c_inden] = key;
}
```

Recursion:

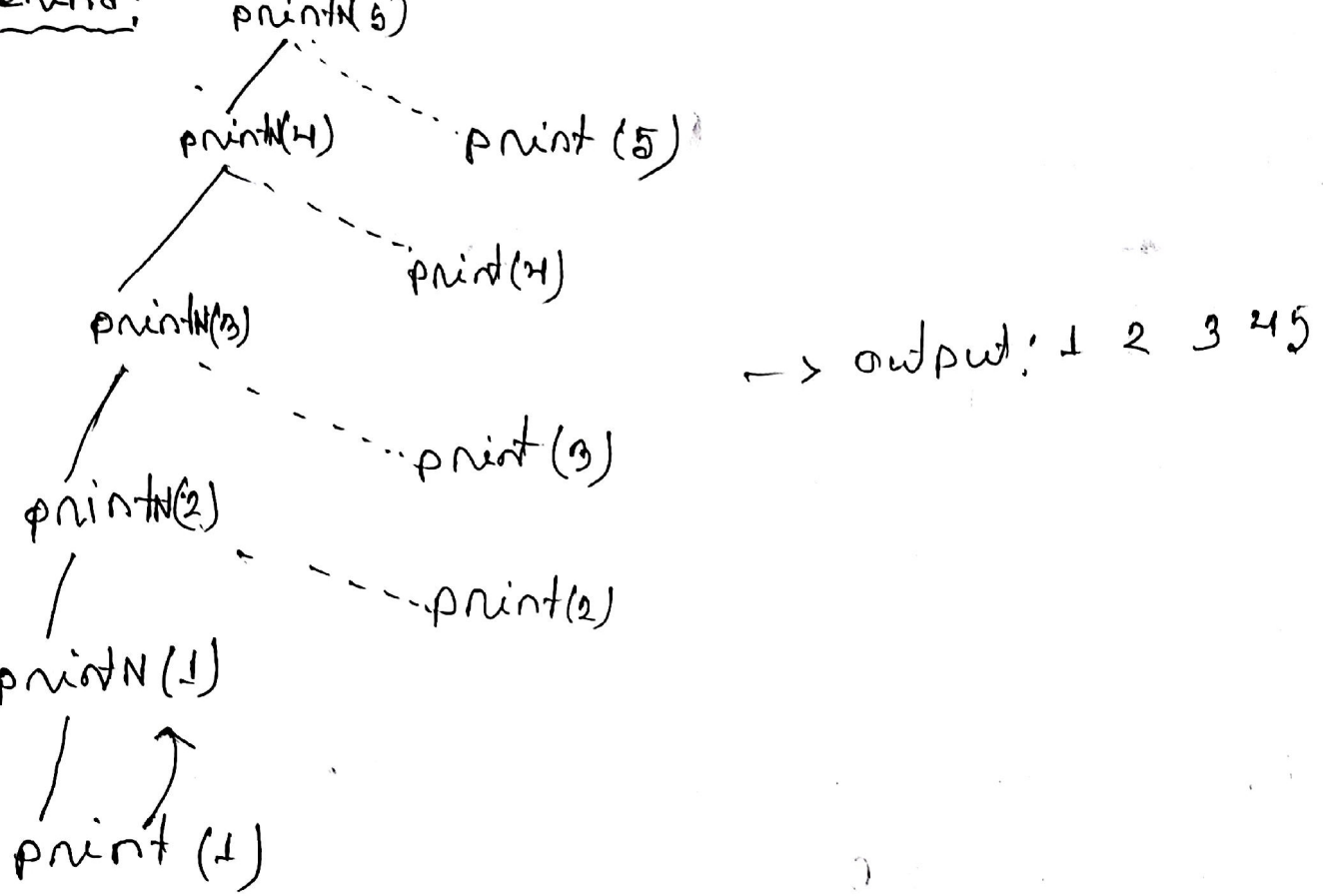
*Sudocode:

```
void printN(int n){
    if(n==1){
        printf("%d", n)
    }
    else{
        printN(n-1);
        printf("%d", n)
    }
```

→ terminating condition/ base case

→ recursive call

*Behind:



→ output: 1 2 3 4 5

ex: Factorial

$n! = 1 \times 2 \times 3 \times \dots \times n$

$0! = 1$

$1! = 1$

** $n! = (n-1)! \times n$

if $n=5$; $= 4! \times 5$

```
int factorial(int n){
 if(n<=1)
     return 1;
 else{
     int friend = factorial(n-1)
     int final = friend*n
     }return final;
 }
```

ex: int power(int a, int b){
if(b==0)
{return 1;
}
else if(b==1){
return a;}
else{
int friend = pow(a,b-1);
int answer = friend * a;
return answer;
}
}

$a^b = b-1 \times a$ (a: dedicate to friend)

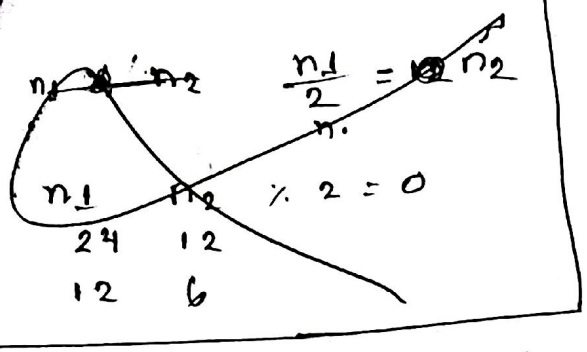
ex: int sum(int n){
if(n<=1) return n;
else{
int friend = sum(n-1);
int answer = friend + n;
return answer;
}
}

1+2+3.....+n

ex: int sumofDigits(int n){
if(n<=9 && n>=0)
return n;
else{
int last_digit = n%10;
int n_without_last_digit = n/10;
int friend = sumofdigit(n_without_last_digit);
in answer = friend + last digit;

27 5 = 2+7+5 (7: dedicated)

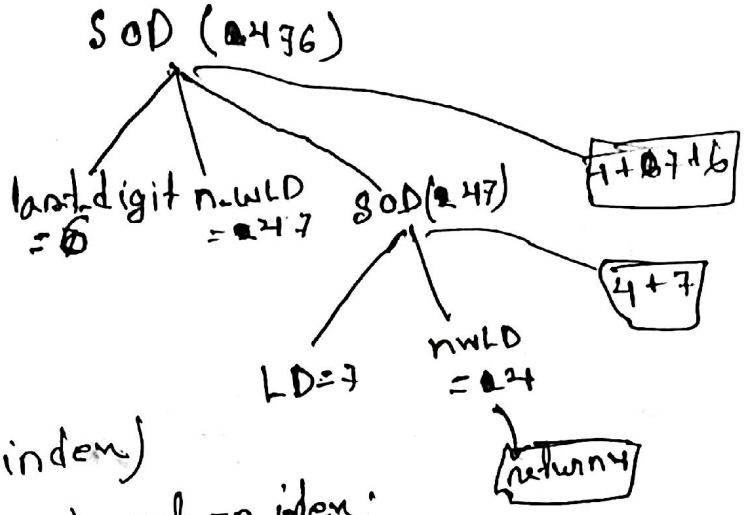
remain =



return answer;

}

}



ex:

```
int gcd_recursive (int a, int b, int index)
{ if(a%. index == 0 && b%index==0) return index;
  else
  { return gcd_recursive (a, b, index-1) ;
  }
}
```

Search

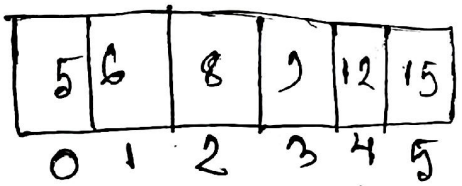
```
int linearSearch (int arr[], int n, int key) {
    for(int i= 0; i<n ; i++) { ⟶ n+1
        if(arr[i] == key) ⟶ n
            return i; ⟶ 0
    }
    return -1 ; ⟶ 1
```

$T(n) = n+1+n+0+1$

$= 2n+2$

i = temp ;

*binary search করতে হলে array অবশ্যই sorted থাকতে হবে।



Low = 0, high = 5 ; mid = $\left\lfloor \frac{0+5}{2} \right\rfloor = 2$
= 2

```
if (arr[mid] == key)
    return mid;
else if key > arr[mid]
    Low = mid + 1
else: high = mid - 1
```

*Code:

```
int binarySearch(int arr[], int n,
                 int key)
{ int low = 0;
  int high = n-1;
  while (low <= high) {
  int mid = (low + high) / 2;
  if (arr[mid] == key)
   { low = mid + 1;
   }
  else high = mid - 1;
 }
 return -1;
}
```

```
int 2^32-1
Low = 2^32-1
high = 2^32-1
mid = low + ⌊(high-low)/2⌋
    = ?
```