Agenda

i) Linear search to find K in A[]

ii) Binary search to find K in A[]

iii) First occurrence of K in array

iv) Floor of K in array

v) Local minima

Q.1 Given an A[], find if K is present or not.

A =	2	9	8	17	42	1
	0	1	2	3	4	5

K = 13 ⇒ -1

K = 17 ⇒ 3

→ apply Linear search

```
int    search ( int [] A, int K ) {

       int n= A.length;

       for (int i=0; i<n; i++) {
       |
       |       if (A[i] == K) {
       |
       |            return i;
       |
       |       }
       }

       return -1;
}
```

TC: O(n)

Searching becomes easy when data is organised.

(Dictionary ...)

Q. 2 Given an sorted array, find if k is present or not.

A =	2	9	13	15	19	24	31	48	52
	0	1	2	3	4	5	6	7	8

k = 15 => 3

k = 48 => 7

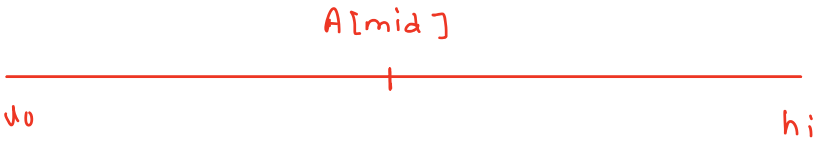
k = 20 => -1

Binary search

K = 13

2	9	$\boxed{13}$	15	19	24	31	48	52
0	1	2	3	4	5	6	7	8
		lo	hi					

lo	hi	mid
0	8	4
0	3	1
2	3	2



Binary search

$$\begin{cases} A[mid] == k & \Rightarrow \text{got the ans} \\ A[mid] < k & \Rightarrow \text{discard left side} \quad lo = mid + 1 \\ A[mid] > k & \Rightarrow \text{discard right side} \quad hi = mid - 1 \end{cases}$$

```
int   search (int [] A, int k) {

   int n = A.length;

   int lo = 0, hi = n-1;

   while ( lo <= hi )   {

       int mid = (lo+hi)/2;

       if (A[mid] == k) {

            return mid;

       }
       else if (A[mid] < k) {
            // discard left side
            lo = mid+1;
       }
       else if (A[mid] > k) {
            // discard right side
            hi = mid-1;

       }
   }

   return -1;

}
```

k = 19

A =	2	4	6	7	10	15	19	21
	0	1	2	3	4	5	6	7
							lo	hi
							m	

=> ans: 6

k = 21

A =	2	4	6	7	10	15	19	21
	0	1	2	3	4	5	6	7
								hi
								lo
								m

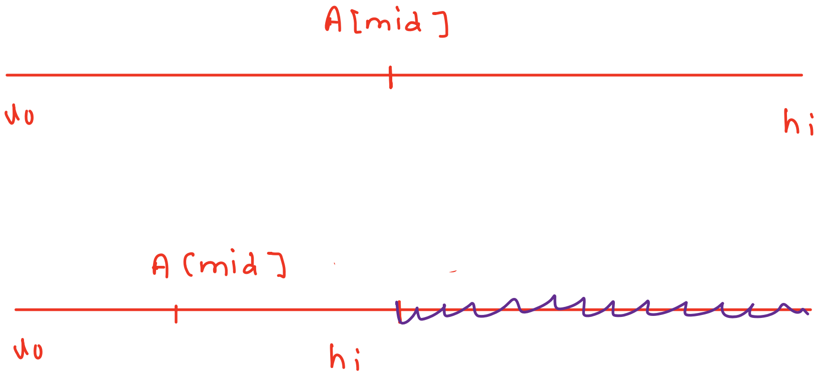
=> ans: 7

k = 5

A =	2	4	6	7	10	15	19	21
	0	1	2	3	4	5	6	7
		hi						
			lo					
			m					

get out of loop

ans => -1



$$\Rightarrow n \rightarrow \frac{n}{2} \rightarrow \frac{n}{4} \cdots 1$$

TC : $O(\log_2 n)$

$$10^5 \approx 2^{16}$$

$$\log_2(10^5) = \log_2(2^{16}) = 16$$

*** In an array of 10^5 elements we can find K in just 16-17 opr.

Q.3 Given a sorted A[], find first occurrence of K.

A =														
	2	2	3	4	5	5	5	7	7	8	9	12	19	20
	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Slight modification in Binary search:

=> A[mid] == K

keep on searching in left side

K=3, ans => 2

K=5, ans => 4

K=10, ans => -1

```
int search (int [] A, int K) {
  int n = A.length;
  int lo = 0, hi = n-1, ans = -1;
  while (lo <= hi) {
     int mid = (lo+hi)/2;
     if (A[mid] == K) {
        ans = mid;
        hi = mid-1;
     }
     else if (A[mid] < K) {
        // discard left side
        lo = mid+1;
     }
     else if (A[mid] > K) {
        // discard right side
        hi = mid-1;
     }
  }
  return ans;
}
```

K = 5

A =									
	2	2	5	5	5	7	9	12	12
	0	1	2	3	4	5	6	7	8
		hi	lo						

ans = ~~-1~~ ~~4~~ 2

K = 5

A =									
	2	5	5	5	5	5	5	5	10
	0	1	2	3	4	5	6	7	8
		lo							
	hi								
	m								

ans = ~~-1~~ ~~4~~ 1

Q-4 Given a sorted array, find floor of K in the array.

floor (K) => max of all the values which are <= K

A =	12	19	21	25	28	32	35	38	42	51
	0	1	2	3	4	5	6	7	8	9

K	floor
25	25
26	25
24	21
37	35

floor(K) = { K if K is present in array; just smaller than K if K is not present in array

K = 24

A =	12	19	21	25	28	32	35	38	42	51
	0	1	2	3	4	5	6	7	8	9
			hi							
				lo						
				m						

ans = ~~19~~ 21

```
if (A[mid] <= K) {              else {
        ans = A[mid];                   hi = mid-1;
        lo = mid+1;             }
}
```

```
int   floor ( int [] A, int  k ) {
    int  n= A. length;
    int  lo= 0, hi= n-1, ans= -1;
    while (lo <= hi ) {
        int  mid= (lo+hi) / 2;
        if (A [mid] <= k ) {
            ans= A [mid];
            lo = mid+1;
        }
        else {
            hi = mid-1;
        }
    }
    return ans;
}
```

k = 19

A =	15	18	21	25	31	38	46	48
	0	1	2	3	4	5	6	7
		hi						
			lo					
			m					

ans = ~~-1~~ 18

k = 50

A =	15	18	21	25	31	38	46	48
	0	1	2	3	4	5	6	7
								hi
								lo
								m

ans = ~~-1~~ ~~36~~ ~~38~~ ~~46~~ 48

Q.5 Local minima

Given an array, find any local minima. Local minima is the element smaller than or equals to both of its neighbours. Corner elements will have only one neighbour. { Array might have duplicates }

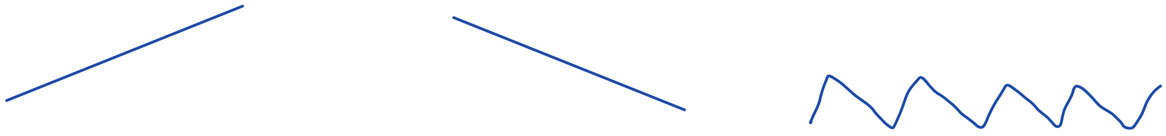
A = 12 > $\boxed{10}$ < 15 20 ans : 10

A = 12 10 9 > $\boxed{7}$ < 15 ans : 7

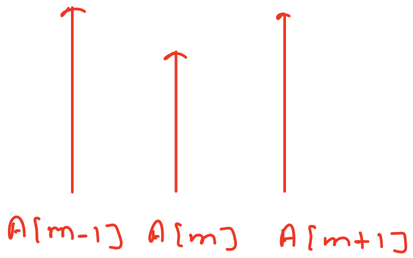
A = 12 15 17 19 8 20 ans : 8 or 12

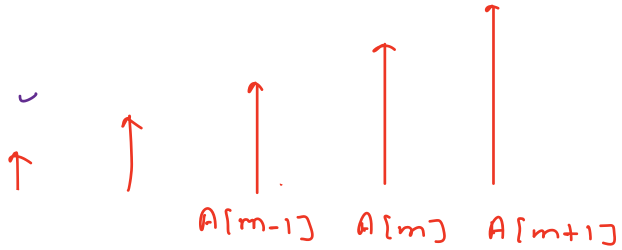
any one of them.

If array contains distinct values, local minima will definitely exists.



Find local minima using binary search :





20	18	22	25	28	16	5
0	1	2	3	4	5	6
lo	m	hi				

5	5	5	5	5	3	2
lo						hi

ans = 5

(considering equality also)

$\boxed{18}$ 20 22 25 28 16 19

Corner cases

```
int   local-minima( int [] A) {
      int  n= A.length;
      if (A[0] <= A[1]) {
            return A[0];
      }
      else if ( A[n-1] <= A[n-2]) {
            return A[n-1];

      }
      int lo = 1 ,   hi = n-2;
      while (lo <= hi ) {
            int  m= ( lo + hi ) / 2 ;
            if ( A[m] <= A[m-1]   && A[m] <= A[m+1]) {
                  return A[m];
            }
            else if ( A[m-1] < A[m]) {
                  hi= m-1;
            }
            else {
                  lo = m+1;
            }
      }
      return -1;
}
```

```
while (lo <= hi) {
    int m = (lo + hi) / 2;
    if (A[m] <= A[m-1]  && A[m] <= A[m+1]) {
        return A[m];
    }
    else if (A[m-1] < A[m]) {
        hi = m-1;
    }
    else {
        lo = m+1;
    }
}
```

A =	9	7	5	4	3	$\boxed{2}$	8
	0	1	2	3	4	5	6
						hi lo m	

A =	11	$\boxed{5}$	6	7	8	5	6	7	8	5	6	7	8	5	6	7	8
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
		lo hi m															