## -> LOWER BOUND & UPPER BOUND

```
INT N;
CIN >> N;    // 6

INT ARR[N];    // 4 5 5 25 7 8
FOR (INT i=0;  i < N;  i++)
{
    CIN >> ARR[i];
}
```

-> To make lower bound & upper bound work efficiently in O(log(n)) operation we have to give it a sorted array or sorted vector.

```
SORT (ARR, ARR+N);
```

~~FOR (INT~~

-> In case of a lower bound returns a pointer. If the element which we are ~~by~~ trying to find is present then lower bound returns that element only, or else it returns its next greater element. If

the element which we are trying to find is higher than maximum no of array, it will return next to last pointer, ARR+N (and if we try to print that it will return a garbage value. So, it is always better to use an if check for that)

→ LOWER_BOUND (STARTING_POINTER, ENDING_POINTER, ELEMENT_TO_FIND); // SYNTAX

```
INT *PTR = LOWER_BOUND (ARR, ARR+N, 5);
  // 5 (element which is present)
INT *PTR2 = LOWER_BOUND (ARR, ARR+N, 6);
  // 7 (element which is not present)
INT *PTR3 = LOWER_BOUND (ARR, ARR+N, 26);
  // ARR+N (element which exceeds the maximum value)

IF (PTR == ARR+N)
{
    COUT << "NOT FOUND" << "\n";
}
ELSE
{
    COUT << *PTR << "\n";
}
```

→ In case of an array upper bound returns a pointer. Upper bound always its next greater element. If the element which we are trying to find its is higher than

maximum no of array it will return next to last pointer, ARR+N (And if we try to print that it will return a garbage value, so, it is always better to use a if check for that).

→ UPPER_BOUND (STARTING_POINTER, ENDING_POINTER, ELEMENT_TO_FIND_NEXT_GREATER_OF); // SYNTAX

INT *PTR4 = UPPER_BOUND (ARR, ARR+N, 5); // 7 (element which is present)

```
IF (PTR4 == ARR+N)
{
    COUT << "NOT FOUND" << "\n";
}
ELSE
{
    COUT << *PTR4 << "\n";
}
```

→ Now, in case of vector upper bound and lower bound returns iterator in place of pointer

→ eg:

```
SORT (V.BEGIN(), V.END);
AUTO IT = LOWER_BOUND (V.BEGIN(), V.END(), 5);
AUTO IT = UPPER_BOUND (V.BEGIN(), V.END(), 5);

IF (IT == A.END())
{
    COUT << "NOT FOUND" << "\n";
}
```

```
ELSE
{
    COUT << *IT << "\n";
}
```

→ In case of sets and maps, We should not use upper bound and lower bound like this:

```
AUTO IT = LOWER_BOUND(S.BEGIN(), S.END(), 5);
// It will give TLE on bigger values, T.C = O(n)
```

Instead use like this:

```
AUTO IT = S.LOWER_BOUND(5); // T.C = O(log(n))
```