*** 이진 탐색 알고리즘**

조건: 정렬

1. 양 끝 원소의 인덱스(low, high)를 이용하여 중간 위치의 인덱스(mid)를 구한다.
2. mid 인덱스의 값(중간값)과 키 값을 비교한다.
3. 키 값이 중간값보다 크면 오른쪽의 절반을 선택한다. → **low = mid + 1 / 중간값은 제외**
4. 키 값이 중간값보다 작으면 왼쪽의 절반을 선택한다. → **high = mid − 1 / 중간값은 제외**
5. 1~4를 반복하고 low = high일 때의 값이 키 값이다.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

1. low = 0, high = 14 → mid = 7
- 키 값인 C와 mid = 7인 H와 비교한다.
- C < H이므로 왼쪽의 절반을 선택한다. → high = mid - 1

2. low = 0, high = 6 → mid = 3
- 키 값인 C와 mid = 3인 D와 비교한다.
- C < D이므로 왼쪽의 절반을 선택한다. → high = mid - 1

3. low = 0, high = 2 → mid = 1
- 키 값인 C와 mid = 1인 B와 비교한다.
- C > B이므로 오른쪽의 절반을 선택한다. → low = mid + 1

4. low = 2, high = 2 → low = high이므로 종료
- 따라서, 4번 비교한다

[반복문을 활용한 이진탐색 알고리즘]

```
int binarySearch (int arr[], int low, int high, int key) {
  while (low <= high) {
    int mid = low + (high-low) / 2;

    if (arr[mid] == key) // 종료 조건1 검색 성공.
      return mid;
    else if (arr[mid] > key)  //왼쪽절반선택하는것
      high = mid - 1;
    else    //오른쪽절반선택하는것
      low = mid + 1;
  }
  return -1 ; // 종료 조건2(값없음) 검색 실패.
}
```

1/2 1/2 1/2.....

((1/2)^k)*n= 1

big0 logn

[과제] 앞서 설명한 이진탐색코드를 재귀적으로 다시 표현해보기

[재귀함수를 활용한 이진탐색 알고리즘]

```
int binarySearch (int arr[], int low, int high, int key) {

  if (low > high)
    return -1;

  int mid = low + (high-low)/2;

  if (arr[mid] == key)
    return mid;

  else if (arr[mid] > key)

      ?

  else

      ?

}
```