Problem 1 | Linear and binary search Analyze and compare

Linear search:

```
int linSrch (int a[], int n, int val) {
    for (int indx = 0; indx < n; indx++) {
        if (val == a[indx])
            return indx;
    }
    return -1;
}
```

let's consider

O_b = operation before for Loop

O_i = Operation inside for loop

POs = operations inside the if-condition

$$O_b + \sum_{indx=0}^{n-1} (O_i + POs)$$

O, operations which equates $T(0)$ → clock cycles.

We know $\sum_{j=x}^{y} 1 = (y-x)+1$

let $O_i + POs = O_{is}$

$O_b + ((n-1)-0+1)\, O_{is}$

$= O_b + n O_{is}$

$f(n)$ is a first order polynomial

$= \boxed{c'n + c}$ Where $c' = O_{is}$ $c = O_b$

→ it is $O(n)$

Binary search :

```
int lowEnd = 0;
int highEnd = 0;
do { int mid = (lowEnd + highEnd)/2;
    if (val == a[mid]) return mid
    else if (val > a[mid]) lowEnd = mid+1;
    else highEnd = mid-1;
} while (lowEnd <= highEnd);
```

let's consider,
Operation before do-while loop = O_b
Operation inside do-while loop = O_d
operation inside if-statement = PO_s
operation inside else-if statement = PO_e
operation inside else statement = PO_1

since in binary-search algorithm, each step search space becomes half for total n-elements of a list

$n \to \frac{n}{2} \to \frac{n}{4} \to \frac{n}{8} \dots \frac{n}{2^i}$, we can write the series as $\frac{n}{2^0} \to \frac{n}{2^1} \to \frac{n}{2^2} \to \frac{n}{2^3} \to \dots \frac{n}{2^i}$

where i is the total number of steps or iterations and after i-steps the search space is reduced to 1

$$\frac{n}{2^i} = 1 \Rightarrow n = 2^i$$
$$\Rightarrow i = \log_2 n$$

$$O_{dsel} = O_d + PO_s + PO_e + PO_1$$

so, for binary search, we can write

$O_b + O_{dsel} \log_2 n \to$ operations, O which equates $T(O) \to$ clock cycles

$= \boxed{c' \log_2 n + c;}$ $c' = O_{dsel}$, $c = O_b$

it is $O(\log n)$