No. :

Date. :

Tugas Pendahuluan

1) Algoritma Pencarian nilai ekstrem
↳ menentukan nilai maksimum (tertinggi) atau minimum (terendah) pada kumpulan data

2.) Algoritma Pencarian yang Terurut Acak. (Sequential Search)
↳ Algoritma pencarian ini digunakan untuk menemukan nilai tertentu pada kumpulan data yang terurut acak. Algoritma akan berhenti ketika kondisi yang dicari sudah ditemukan

3.) Algoritma Pencarian pada Array Terurut (Binary Search)
↳ Algoritma pencarian yang dipelajari sebelumnya dilakukan pada tabel acak. Walaupun bisa diaplikasikan ke tabel terurut, tapi tidak optimal. Binary search membagi kumpulan data menjadi 2 bagian, secara berbeda untuk mengecek setiap sub bagian tersebut. Pencarian akan berakhir ketika kondisi yang dicari terpenuhi.

Perbedaan Sequential Search & Binary Search

Sequential search : Mencari dari awal sampai akhir.

Binary Search = membagi 2 pencarian secara seluruh. dan mencari di sub-array nya

no2.go > sequentialsearch

```go
package main
type arr [998]int

func sequentialsearch(T arr, n int, x int ) int{
    var ketemu bool
    var k int
    ketemu = false
    k = 0
    for !ketemu && k < n {
        ketemu = T[k] == x
        if ketemu {
            break
        }
        k += 1

    }
    if k > n {
        k = -1
    }
    return k
}
```

```go
package main
type tabInt [999]int
func ascending(tab tabInt, n, x int) int {
    var left, right, mid int
    var found int
    left = 1
    right = n
    found = -1
    for left <= right && found == -1 {
        mid = (left + right) / 2
        if x < tab[mid] {
            right = mid - 1
        } else if x > tab[mid] {
            left = mid + 1
        } else {
            found = mid
        }
    }
    return found
}
func descending(tab tabInt, n, x int) int {
    var left, right, mid int
    var found int
    left = 1
    right = n
    found = -1
    for left <= right && found == -1 {
        mid = (left + right) / 2
        if x > tab[mid] {
            right = mid - 1
        } else if x < tab[mid] {
            left = mid + 1
        } else {
            found = mid
        }
    }
    return found
```

no4.go > exist

```go
package main
import "fmt"
type set [2022]int
func main() {
    var s1, s2, s3 set // s = array, n = bilangan int
    var n1, n2, n3 int
    inputSet(&s1, &n1)
    inputSet(&s2, &n2)
    findIntersection(s1, s2, n1, n2, &s3, &n3)
    printSet(s3, n3)
}
func exist(T set, n int, val int) bool {
    var state bool = false
    for i := 0; i < n; i++ {
        if T[i] == val {
            state = true
        }
    }
    return state
}
func inputSet(T *set, n *int) {
    for i := 0; i < len(*T); i++ {
        fmt.Scan(&*n)
        if exist(*T, i, *n) {
            break
        }
        T[i] = *n
    }
}
func findIntersection(T1, T2 set, n, m int, T3 *set, h *int) {
    for _, x := range T1 {
        for i, y := range T2 {
            if x == y && x != 0 {
                T3[i] = x
            }
        }
    }
```

```
35                  }
36              }
37          }
38      }
39      func printSet(T set, n int) {
40          for i := 0; i < len(T); i++ {
41              if T[i] != 0 {
42                  fmt.Print(T[i], " ")
43              }
44          }
45      }
```