1. **Fungsi** adalah baris instruksi yang memiliki nilai kembalian sedangkan **Prosedur** adalah suatu hal yang berisi instruksi yang bisa dijalankan namun tidak memiliki nilai kembalian seperti fungsi.
2. **Fungsi Rekursif** adalah sebuah Method (fungsi /Prosedur) yang bisa memanggil dirinya sendiri dan berguna dalam memecahkan masalah tertentu dam membutuhkan kondisi tertentu agar bisa berhenti.
3. 1.**sequencial search** adalah metode pencarian dengan membandingkan data data yang ada dalam kumpulan, mulai dari elemen pertama sampai elemen ditemukan,atau sampai akhir elemen

   2.**binary search** adalah metode pencarian data yang terurut dengan membandingkan harga yang dicari dengan harga elemen tengah atau titik tengah, jika sama berarti ketemu, jika titik tengah lebih besar mencari pada bagian sebelah kiri
4. 1. **Bubble sort**, Proses pengurutannya dilakukan dengan cara membandingkan masing-masing nilai dalam suatu list secara berpasangan, kemudian pertukaran akan terjadi jika nilai data ke-[i] lebih kecil daripada data ke-[i-1], sebaliknya jika nilai data ke-[i] lebih besar daripada nilai data ke-[i-1] maka pertukaran tidak akan terjadi dan terus melanjutkan perbandingannya dengan nilai data ke-[i] yang menjadi patokannya.

   2. **Selection sort** adalah metode pencarian yang membandingkan elemen list yang pertama dengan elemen yang berikutnya sampai dengan elemen yang terakhir. Jika ditemukan elemen lain yang lebih kecil dari elemen yang sekarang maka ditandai posisinya dan kemudian akan saling bertukar tempat.

   3.**quick sort** melakukan sorting dengan membagi masalah menjadi sub masalah dan sub masalah dibagi lagi menjadi sub-sub masalah sehingga sorting tersebut menjadi lebih cepat walaupun memakan ruang memori yang besar.

   4.**Shell Sort** adalah algoritma yang cara kerjanya dengan cara membandingkan suatu data lain yang memiliki jarak tertentu sehingga membentuk sebuah sub-list.

   5.**insertion sort** merupakan pengurutan data yang membandingkan dengan dua elemen data pertama, kemudian membandingkan elemenelemen data yang sudah diurutkan, kemudianperbandingan tersebut akan terus diulang hingga tidak ada elemen data yang tersisa.

.

5. #Angka terakhir 8 = [14, 1, 4, 2, 16, 8, 5, 10, 12, 18]

```
def bubbleSort(arr):
    lanjut = True
    while lanjut:
        lanjut = False
        for i in range(0, len(arr)-1):
            if arr[i] > arr[i + 1]:
                nilailebihBesar = arr[i]
                arr[i] = arr[i + 1]
                arr[i + 1] = nilailebihBesar
                lanjut = True
    return arr
arrayBelumUrut = [ 14, 1, 4, 2, 16, 8, 5, 10, 12, 18 ]
print ("arrayBelumUrut", arrayBelumUrut)

arraySudahUrut = bubbleSort(arrayBelumUrut)
print ("arraySudahUrut", arraySudahUrut)
```

6 #Angka terakhir 8 = [14, 1, 4, 2, 16, 8, 5, 10, 12, 18]

```
def insertionSort(arr):
    n = len(arr)
    for i in range(0, n):
        nilaiYangSedangDicek = arr[i]
        j = i - 1
        while j >= 0 and arr[j] > nilaiYangSedangDicek :
            arr[j + 1] = arr[j]
            j = j-1
            arr[j + 1] = nilaiYangSedangDicek
    return arr

arrayBelumUrut = [ 14, 1, 4, 2, 16, 8, 5, 10, 12, 18 ]
print ("arrayBelumUrut", arrayBelumUrut)


arraySudahUrut = insertionSort(arrayBelumUrut)

print ("arraySudahUrut", arraySudahUrut)
```

7. #Angka terakhir 8 = [14, 1, 4, 2, 16, 8, 5, 10, 12, 18]

```
def quickSort(arr):
    n = len(arr)
    if n <=1 :
        return arr
    left = []
    right = []
    equal = []
    pivot = arr [-1]
    for num in arr:
        if num < pivot:
            left.append(num)
        elif num == pivot:
            equal.append(num)
        else:
            right.append(num)
    return quickSort(left)+equal + quickSort(right)

arrayBelumUrut = [ 14, 1, 4, 2, 16, 8, 5, 10, 12, 18 ]
print ("arrayBelumUrut", arrayBelumUrut)

arraySudahUrut = quickSort(arrayBelumUrut)
print ("arraySudahUrut", arraySudahUrut)
```

8. #Angka terakhir 8 = [14, 1, 4, 2, 16, 8, 5, 10, 12, 18]

binarySearch dengan diurutkan terlebih dahulu dengan quickSort

```
def quickSort(arr):
    n = len(arr)
    if n <=1 :
        return arr
    left = []
    right = []
    equal = []
    pivot = arr [-1]
    for num in arr:
        if num < pivot:
            left.append(num)
        elif num == pivot:
            equal.append(num)
        else:
```

```
            right.append(num)
    return quickSort(left)+equal + quickSort(right)

def binarySearch(arr, low, high, x):
    if high >= low:
        mid = (low + high) //2
        if arr[mid] == x:
            return mid
        elif x < arr[mid]:
            return binarySearch(arr, mid + 1, high, x)
    else:
        return -1

arrayBelumUrut = [ 14, 1, 4, 2, 16, 8, 5, 10, 12, 18 ]
print ("arrayBelumUrut", arrayBelumUrut)

arraySudahUrut = quickSort(arrayBelumUrut)
print ("arraySudahUrut", arraySudahUrut)
hasilPencarian = binarySearch(arraySudahUrut, 0, len(arraySudahUrut) -1, 8)
print("hasil pencarian angka 8 berada pada indeks ke ",hasilPencarian)
```