Nama : Deanissa Sherly Sabilla

Kelas : SIB 1B

Absen : 06

-PERTEMUAN 6-

-SORTING-

1. Data = {23,35,14,7,67,89,20}
   Gambarkan proses penyelesaian kasus pengurutan data di atas dengan menggunakan algoritma
   - Bubble Sort untuk pengurutan descending

```
bubbleSort(arr, size)
    for i in range(size - 1)
        for j in range(size - i - 1)
            if arr[j] < arr[j + 1]
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
```

   Penjelasan :
   1) **bubbleSort(arr, size)** : deklarasi dari **bubbleSort** dan dengan dua parameter ‘arr’ (array yang diurutkan) dan ‘size’ (ukuran array)
   2) **for i in range(size - 1)** : loop luar yang berjalan dari 0 sampau ‘size-1’, sehingga elemen akhir tidak perlu dibandingkan lagi dengan iterasi terakhir
   3) **for j in range(size - i - 1)** : loop dalam yang berjalan dari 0 sampai ‘size-i-1’, karena setiap iterasi dari loop luar, elemen akhir akan terurut dan tidak dibandingkan lagi
   4) **if arr[j] < arr[j + 1]** : kondisi memeriksa apakah elemen ini lebih kecil dari elemen berikutnya, jika iya maka kedua elemen akan ditukar posisinya
   5) **arr[j], arr[j + 1] = arr[j + 1], arr[j]** : proses petukaran posisi elemen jika kondisi sebelumnya terpeuhi, dapat menukar nilai ‘arr[j]’ dan ‘arr[j+1]’ dengan satu baris kode

   - Selection Sort untuk pengurutan ascending

```
selectionSort(arr, size)
    for i in range(size - 1)
        minIndex = i
        minValue = arr[i]
        for j in range(i + 1, size)
            if arr[j] < minValue
                minIndex = j
                minValue = arr[j]
        arr[i], arr[minIndex] = arr[minIndex], arr[i]

    return arr
```

   Penjelasan :
   1) **selectionSort(arr, size)** : deklarasi dari **selectionSort** dan dengan dua parameter ‘arr’ (array yang diurutkan) dan ‘size’ (ukuran array)

2) **for i in range(size - 1)** : loop luar yang berjalan dari 0 sampau ‘size-1’, sehingga elemen akhir tidak perlu dibandingkan lagi dengan iterasi terakhir
3) **minIndex = i** : insialisasi ‘minIndex’ dengan ‘i’ menandakan elemen terkecil yang belum diurutkan
4) **minValue = arr[i**] : inisialisasi ‘minValue’ dengan nilai elemen indeks ‘i’, sebagai nilai elemen terkecil yang belum diurutkan
5) **for j in range(i + 1, size)** : loop dalam berjalan dari ‘i+1’ hingga ‘size-1’ mencari elemen terkecil dari sisa array yang belum diurutkan
6) **if arr[j] < minValue** : kondisi apakah nilai elemen dari indeks ‘j’ lebih kecil dari ‘minValue’, jika ya, maka nilai ‘minValue’ diperbarui menjadi nilai ‘arr[j]’ dan ‘minIndex’ diperbarui menjadi ‘j’
7) **arr[i], arr[minIndex] = arr[minIndex], arr[i]** : setelah menemukan elemen terkecil, baris ini bertugas untuk menukar elemen terkecil tersebut dengan elemen pada indeks i, sehingga elemen terkecil diposisikan di awal array yang belum diurutkan
8) **return arr** : seluruh iterasi terakhir, fungsi mengembalikan array yang sudah diurutkan

- Insertion Sort untuk pengurutan descending

```
insertionSort(arr, size)
    for i in range(size - 1)
        temp = arr[i]
        j = i
        while j > 0 and arr[j - 1] < temp
            arr[j] = arr[j - 1]
            j -= 1
        arr[j] = temp

    return arr
```

Penjelasan :

1) **insertionSort(arr, size)** : deklarasi dari **insertionSort** dan dengan dua parameter ‘arr’ (array yang diurutkan) dan ‘size’ (ukuran array)
2) **for i in range(size - 1)** : loop luar yang berjalan dari 0 sampau ‘size-1’, sehingga elemen akhir tidak perlu dibandingkan lagi dengan iterasi terakhir
3) **temp = arr[i]** : memindahkan nilai dari elemen ke – i array ke dalam variabel ‘temp’
4) **j = i** : inisialisasi ‘j’ dengan ‘i’, variabel j akan digunakan untuk menyimpan indeks dari elemen yang di simpan
5) while j > 0 and arr[j - 1] < temp : loop dalam berjalan selama ‘j’ lebih besar dari 0 (indeks belum mencapai awal array) dan nilai elemen sebelumnya (arr[j - 1]) lebih kecil dari temp (nilai elemen yang sedang diproses). Kondisi ini bertujuan untuk mencari posisi yang tepat untuk memasukkan temp dalam mengurutkan array secara descending.
6) **arr[j] = arr[j - 1]** : memindahkan nilai elemen sebelumnya ke posisi saat ini ‘j’

7) **j -= 1** : mengurangi nilai ‘j’ untuk memeriksa elemen sebelumnya dalam array
8) **arr[j] = temp** : setelah menemukan posisi yang tepat untuk temp, nilai temp dimasukkan ke posisi tersebut
9) **return arr** : setelah seluruh iterasi selesai, fungsi mengembalikan array yang sudah diurutkan

2. Jelaskan tindakan yang dilakukan pada algoritma Bubble Sort dan Selection Sort jika menemukan elemen data yang sama nilainya!
   Contoh = {22,33,45,17,33}
   - **Bubble Sort**
     Ketika Bubble Sort menemukan 2 elemen yang memiliki nilai yang sama, algoritma tersebut tidak akan melakukan pertukaran posisi untuk elemen tersebut. Oleh karena itu, elemen-elemen dengan nilai yang sama akan tetap berada di posisi yang sama seperti di awalnya setelah proses pengurutan selesai.

     Dalam contoh di atas, nilai 33 muncul 2 kali. Setelah proses pengurutan dengan Bubble Sort, posisi kedua elemen 33 tersebut tidak akan berubah.
   - **Selection Sort**
     Ketika Selection Sort menemukan 2 elemen yang memiliki nilai yang sama, algoritma tersebut akan tetap memilih salah satu elemen untuk dipindahkan ke posisi yang sesuai selama proses pengurutan. Sehingga, elemen-elemen dengan nilai yang sama akan tetap berada di posisi yang sama seperti di awalnya setelah proses pengurutan selesai.

     Dalam contoh di atas, nilai 33 muncul 2 kali dalam array. Ketika Selection Sort sedang memilih elemen terkecil untuk dipindahkan ke posisi yang sesuai, algoritma ini akan memilih salah satu dari elemen 33 tersebut. Namun, posisi kedua elemen 33 tersebut tidak akan berubah setelah proses pengurutan selesai.