1. Buat modifikasi algoritma bubble sort. Jika algoritma bubble sort, pointer bergerak dari kiri ke kanan (menuju index terakhir), sedemikian hingga data terbesar berada pada index terakhir. Modifikasi yang dilakukan adalah, setelah data terbesar terletak pada index terakhir. Pointer bergerak lagi dari kanan ke kiri untuk membawa data terkecil ke index paling awal .

   Berikut contoh jumlah iterasi yang dieksekusi jika menggunakan algoritma bubble sort konvensional

```
b=[10,2,5,8,1,20,7,3,4]
bubbleSort(b)
```

```
Bubble Sort
Data Awal [10, 2, 5, 8, 1, 20, 7, 3, 4]
[2, 5, 8, 1, 10, 7, 3, 4, 20]
[2, 5, 1, 8, 7, 3, 4, 10, 20]
[2, 1, 5, 7, 3, 4, 8, 10, 20]
[1, 2, 5, 3, 4, 7, 8, 10, 20]
[1, 2, 3, 4, 5, 7, 8, 10, 20]
[1, 2, 3, 4, 5, 7, 8, 10, 20]
[1, 2, 3, 4, 5, 7, 8, 10, 20]
[1, 2, 3, 4, 5, 7, 8, 10, 20]
[1, 2, 3, 4, 5, 7, 8, 10, 20]
Data urut- [1, 2, 3, 4, 5, 7, 8, 10, 20]
```

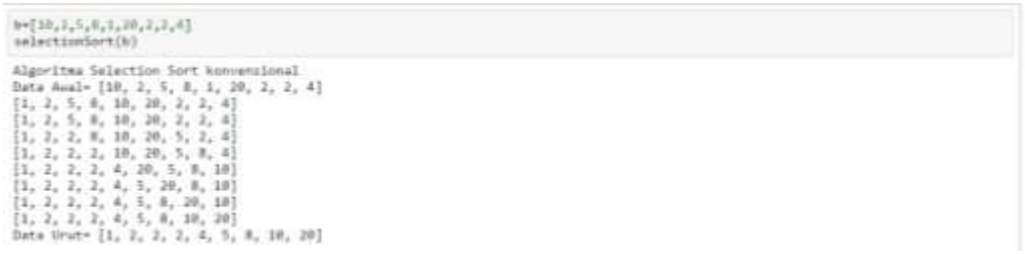
   Berikut contoh jumlah iterasi yang dilakukan jika menggunakan modifikasi algoritma bubble sort. Jumlah iterasi berkurang hampir 50% dari jumlah iterasi algoritma bubble sort konvensional

```
b=[10,2,5,8,1,20,7,3,4]
alg.bubbleSort3(b)
```

```
Modified Bubble Sort
Data Awal = [10, 2, 5, 8, 1, 20, 7, 3, 4]
[1, 2, 5, 8, 3, 10, 7, 4, 20]
[1, 2, 3, 5, 4, 8, 7, 10, 20]
[1, 2, 3, 4, 5, 7, 8, 10, 20]
[1, 2, 3, 4, 5, 7, 8, 10, 20]
sortedData= [1, 2, 3, 4, 5, 7, 8, 10, 20]
```

2. Buat modifikasi algoritma selection sort. Jika algoritma selection sort konvensional memilih bilangan paling kecil terlebih dahulu, kemudian bilangan tersebut diletakkan di posisi paling kiri dari data, maka buatlah modifikasi algoritma tersebut dengan cara, tambahkan beberapa perintah agar pada iterasi tersebut tidak hanya memilih data paling kecil dan memindahkan data paling kecil di posisi paling kiri, akan tetapi juga memilih data paling besar dan memindahkan data terbesar tersebut di posisi paling kanan.

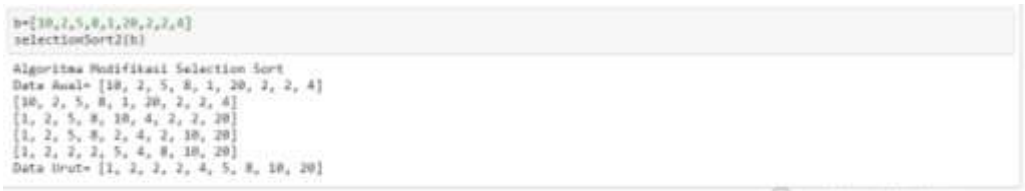
   Berikut contoh jumlah iterasi yang dilakukan dengan menggunakan algoritma selection sort konvensional :



```
b=[10,2,5,8,1,20,2,2,4]
selectionSort(b)

Algoritma Selection Sort konvensional
Data Awal= [10, 2, 5, 8, 1, 20, 2, 2, 4]
[1, 2, 5, 8, 10, 20, 2, 2, 4]
[1, 2, 5, 8, 10, 20, 2, 2, 4]
[1, 2, 2, 8, 10, 20, 5, 2, 4]
[1, 2, 2, 2, 10, 20, 5, 8, 4]
[1, 2, 2, 2, 4, 20, 5, 8, 10]
[1, 2, 2, 2, 4, 5, 20, 8, 10]
[1, 2, 2, 2, 4, 5, 8, 20, 10]
[1, 2, 2, 2, 4, 5, 8, 10, 20]
Data Urut= [1, 2, 2, 2, 4, 5, 8, 10, 20]
```

   Berikut contoh jumlah iterasi yang dilakukan dengan menggunakan algoritma selection sort yang sudah dimodifikasi.



```
b=[10,2,5,8,1,20,2,2,4]
selectionSort2(b)

Algoritma Modifikasi Selection Sort
Data Awal= [10, 2, 5, 8, 1, 20, 2, 2, 4]
[10, 2, 5, 8, 1, 20, 2, 2, 4]
[1, 2, 5, 8, 10, 4, 2, 2, 20]
[1, 2, 5, 8, 2, 4, 2, 10, 20]
[1, 2, 2, 2, 5, 4, 8, 10, 20]
Data Urut= [1, 2, 2, 2, 4, 5, 8, 10, 20]
```

   Dapat dilihat bahwa jumlah iterasi berkurang hampir 50% dari jumlah iterasi algoritma selection sort konvensional

3. Generate 1000 bilangan acak, urutkan dengan menggunakan selection sort konvensional dan modifikasi. Bandingkan waktu yang diperlukan antara dua algoritma tersebut untuk mengurutkan 1000 bilangan acak.
   Berikut contoh hasil perhitungan waktu antara kedua algoritma tersebut

```
Algoritma Selection Sort konvensional
0.078125
Algoritma Modifikasi Selection Sort
0.046875
```