Lab Tutorial 4

Berbeda dengan DDP1 maupun DDP2, pada kuliah SDA ini kita menggunakan *Automatic Grader* untuk mengecek *correctness* dari program anda menggunakan beberapa kasus uji (test case), anda cukup mengunggah pekerjaan anda ke slot yang sudah di sediakan di Scele.

Perhatikan bahwa pada bagian atas soal Lab ini terdapat keterangan nama berkas. Berkas yang Anda unggah ke Scele **harus mengikuti peraturan penamaan ini**. Berbeda dengan kuliah DDP, kami juga akan menilai efektivitas program Anda dari segi waktu dan memori. Batas waktu eksekusi program adalah 3 detik, artinya program Anda harus berjalan paling lama 3 detik di server grader Scele untuk setiap kasus uji. Jika lebih dari itu, maka pada kasus uji tersebut program Anda dinyatakan salah. Batas memori yang digunakan oleh program Anda adalah 256 MB. Jika Anda menggunakan struktur data yang besar, contohnya membuat array dengan ukuran yang sangat besar, maka program Anda akan dinyatakan salah.

Berbeda dengan minggu - minggu sebelum nya, pada lab minggu ini akan berisi tutorial dan soal. Tutorial akan berisi materi dan penjelasan mengenai topik lab minggu ini, sementara latihan soal seperti minggu sebelum nya berisi soal - soal untuk memperdalam pemahaman mengenai topik yang dibahas.

Problem Setter : FSP

Lab Tutorial 4A

Quick Sort

Nama berkas kode sumber	:	SDA<npm>L4A.java
Batas waktu eksekusi program	:	3 detik / kasus uji
Batas memori program	:	256 MB / kasus uji

Quick sort adalah algoritma sorting yang berdasarkan pembandingan dengan metode divide-and-conquer. Disebut *Quick Sort*, karena algoritma *quick sort* mengurutkan dengan sangat cepat, namun algoritma ini sangat kompleks dan diproses secara rekursif. Sangat memungkinkan untuk menulis algoritma yang lebih cepat untuk beberapa kasus khusus, namun untuk kasus umum, sampai saat ini tidak ada yang lebih cepat dibandingkan algoritma *quick sort*. Untuk memulai iterasi pengurutan, pertama-tama sebuah elemen dipilih dari data sebagai pivot, kemudian elemen-elemen data akan diurutkan diatur sedemikian rupa.

Berikut adalah implementasi kode java untuk *quick sort* untuk nilai terkecil ke nilai terbesar. Pada contoh ini kita akan mengambil nilai *pivot* yaitu **n / 2**.

```
public static int[] quicksort(int[] arr, int low, int high) {
        int i = low, j = high;
        // TODO menentukan pivot yaitu data tengah dari array
        int pivot =

        // Dibagi menjadi 2 bagian (kiri pivot & kanan pivot)
        while (i <= j) {
            // Jika nilai i sekarang lebih kecil dari pivot, maka i
            // menjadi setelahnya dari bagian kiri
            while (numbers[i] < pivot) {
                // TODO increment i.
            }
```

Problem Setter : FSP

```
            // Jika nilai j sekarang lebih besar dari pivot, maka j
            // menjadi setelahnya dari bagian kanan
            while (numbers[j] > pivot) {
                // TODO decrement j.
            }

            // Jika kita sudah menemukan nilai dari bagian kiri yang
            // lebih kecil dari pivot dan nilai dari bagian kanan
            // yang lebih besar dari pivot, maka tukar nilai
            // tersebut.
            // Lalu geser i dan j.
            if (i <= j) {
                // TODO swap nilai dari i dan j
                // TODO increment i
                // TODO decrement j
            }
        }
        // Rekursif
        if (low < j)
            // TODO jalankan method ini dari low hingga j
        if (i < high)
            // TODO jalankan method ini dari i hingga high
    }
```

Ilustrasi *quick sort* dapat dilihat pada animasi di link berikut

https://upload.wikimedia.org/wikipedia/commons/9/9c/Quicksort-example.gif

Latihan

Problem Setter : FSP

Terapkan algoritma *quick sort* seperti yang sudah dibahas diatas untuk mengurutkan angka **terurut naik**. Pada soal ini anda diharapkan untuk mengimplementasikan algoritma quick sort sendiri. Bagi mahasiswa yang menggunakan algoritma sorting bawaan *java* atau sumber lainnya akan mendapatkan nilai 0.

Format Masukan

Baris pertama dari input adalah **N** yang menyatakan jumlah angka yang harus diurutkan.

Baris kedua berisi angka berjumlah **N** yang dipisahkan spasi.

Format Keluaran

Hasil pengurutan masukan yang dipisahkan dengan spasi.

Constraint

- $1 \leq N \leq 10^4$
- $-10^6 \leq \text{angka} \leq 10^6$

Contoh Masukan

```
10
2 7 11 8 22 64 82 12 89 0
```

Contoh Keluaran

```
0 2 7 8 11 12 22 64 82 89
```

Problem Setter : FSP

Lab Tutorial 4B

Yu-Gi-Ahh

Nama berkas kode sumber	:	SDA<npm>L4B.java
Batas waktu eksekusi program	:	3 detik / kasus uji
Batas memori program	:	256 MB / kasus uji

Yu-Gi-Ahh adalah permainan kartu yang sedang populer pada anak-anak zaman sekarang. Salah satunya adalah Wiken, yang merupakan salah satu pemain Yu-Gi-Ahh profesional. Sebagai pemain profesional, ia tidak mau kalah. Sehingga ia selalu berlatih menyusun strategi permainannya saat *weekend*.

Sebagai programmer, strategi yang dilakukan Wiken mempunyai pola tertentu. Ia selalu mengurutkan kartu yang ada di tangannya. Satu kartu Yu-Gi-Ahh **mempunyai nama dan nilai power**. Wiken akan mengurutkan kartunya dari **power yang terbesar hingga yang terkecil, jika ada power yang sama maka selanjutnya akan diurutkan *lexicographic* secara *Ascending***. Selanjutnya, strategi yang digunakan Wiken adalah ***Attack*** dan ***Defense*** yang akan dijelaskan berikutnya.

Format Masukan

Masukan akan sampai ***eof*** (***end of file***).

Satu kartu Yu-Gi-Ahh mempunyai nama dan power, nama terdiri dari **String 1 kata** dan power merupakan bilangan bulat positif ***N.***

Selanjutnya perintah-perintah strategi adalah sebagai berikut.

PERINTAH	**OUTPUT**	**PENJELASAN**
PICK [Nama Kartu] ***N***	[Nama Kartu] dengan power ***N*** diambil	Mengambil satu kartu ke tangan dan Wiken akan **langsung mengurutkan kartu yang ada di tangannya**.
ATTACK	[Nama Kartu] ***N*** dikeluarkan	Jika Attack dipanggil, artinya

Problem Setter : FSP

	Jika tidak ada kartu di tangan maka output **"Tidak bisa melakukan Attack"**	Wiken harus mengeluarkan 1 kartu dengan power terbesarnya. N menunjukkan power dari kartu tersebut.
DEFENSE	[Nama Kartu 1] N_1 dikeluarkan [Nama Kartu 2] N_2 dikeluarkan [Nama Kartu 3] N_3 dikeluarkan Jika kartu tidak mencapai 3 buah maka output **"Tidak bisa melakukan Defense"**	Jika Defense dipanggil, maka Wiken akan mengeluarkan 3 kartu dengan power terendahnya. Urutan pengeluaran kartu adalah dari yang terkecil dari tangannya.
SEE CARD	[Nama Kartu 1] N_1 [Nama Kartu 2] N_2 [Nama Kartu 3] N_3 …… Jika kartu di tangan kosong maka output **"Kartu kosong"**	Melihat semua kartu yang ada di tangan, urutan output adalah secara bagaimana Wiken mengurutkan kartunya.

Format Keluaran

Format keluaran seperti yang dijelaskan di atas.

Constraint

- $1 \leq N \leq 10^4$
- Perintah akan selalu dalam format *Uppercase*
- Nama kartu hanya terdiri dari 1 kata alfanumerik dan dijamin unik
- Power merupakan bilangan bulat positif

Contoh Masukan

```
PICK Jeano 20
```

Problem Setter : FSP

```
PICK Adelle 5
ATTACK
DEFENSE
PICK Bobbo 3
PICK Alamak 5
PICK Yodelei 3
SEE CARD
DEFENSE
```

Contoh Keluaran

```
Jeano dengan power 20  diambil
Adelle dengan power 5 diambil
Jeano 20 dikeluarkan
Tidak bisa melakukan Defense
Bobbo dengan power 3 diambil
Alamak dengan power 5 diambil
Yodelei dengan power 3 diambil
Adelle 5
Alamak 5
Bobbo 3
Yodelei 3
Yodelei 3 dikeluarkan
Bobbo 3 dikeluarkan
Alamak 5 dikeluarkan
```

Catatan

- **Kartu saat awal program adalah kosong**
- **Anda diharuskan menggunakan implementasi *quick sort* pada latihan ini jika tidak maka nilai akan sama dengan 0**

Problem Setter : FSP