Started on	Monday, 20 May 2024, 2:25 PM
State	Finished
Completed on	Wednesday, 5 June 2024, 5:28 AM
Time taken	15 days 15 hours
Grade	**300.00** out of 300.00 (**100**%)

Question **1**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Anda diminta untuk mengimplementasikan kelas PenjurusanTPB. Dalam kelas PenjurusanTPB terdapat kelas (Nested Class):

- Kelas Mahasiswa yang memiliki atribut NIM, Nama, IP, dan UKT. Terdapat konstruktor yang menerima NIM, Nama, IP, dan UKT. Terdapat pula aksesor untuk mengakses NIM, Nama, IP, dan UKT.
- Terdapat kelas MahasiswaComparator yang mengimplementasikan Comparator<Mahasiswa>. Kelas ini akan membandingkan antar-mahasiswa dengan prioritas sebagai berikut:

  1. UKT yang lebih besar akan diprioritaskan.
  2. IP yang lebih besar akan diprioritaskan.
  3. NIM yang lebih kecil akan diprioritaskan. (NIM dipastikan unik)

Selain itu, terdapat method PembangkitanAntrianPrioritas yang menerima List<Mahasiswa> dan mengembalikan PriorityQueue<Mahasiswa> yang sudah terurut berdasarkan prioritas.

Lengkapilah file `PenjurusanTPB.java` dan kumpulkan kembali file `PenjurusanTPB.java` yang telah berisi jawaban Anda.

Docs:
Priority Queue: https://docs.oracle.com/javase/8/docs/api/java/util/PriorityQueue.html
Comparator: https://docs.oracle.com/javase/8/docs/api/java/util/Comparator.html

Java 8

PenjurusanTPB.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	20	Accepted	0.17 sec, 29.92 MB
2	20	Accepted	0.11 sec, 26.80 MB
3	20	Accepted	0.10 sec, 28.30 MB
4	20	Accepted	0.09 sec, 28.10 MB
5	20	Accepted	0.09 sec, 28.34 MB

Question **2**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Queue From Stack

Anda diminta untuk mengimplementasikan Queue (First In First Out) dengan hanya menggunakan 2 stack. Queue yang diimplementasikan diperlukan untuk mendukung fungsi - fungsi dari sebuah queue yaitu push, peek, pop, dan empty.

void push(int x) : Menambah elemen x ke belakang Queue
int pop() : Menghapus elemen dari depan Queue dan mengembalikan elemen tersebut
int peek() : Mengembalikan elemen depan Queue
boolean empty() : Mengembalikan apakah Queue kosong (true) atau tidak (false)

Anda hanya bisa menggunakan operasi standar dari sebuah stack yaitu:
1. push(Integer item) : Menyimpan item ke puncak stack
2. peek() : Melihat item yang berada di puncak stack tanpa mengeluarkannya dari stack
3. pop() : Menghapus item di puncak stack dan mengembalikan item tersebut
4. empty() : Menguji apakah stack kosong

Submit file MyQueue.java

Java 8

MyQueue.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	20	Accepted	0.15 sec, 33.48 MB
2	20	Accepted	0.15 sec, 33.50 MB
3	20	Accepted	0.15 sec, 33.30 MB
4	20	Accepted	0.15 sec, 34.70 MB
5	20	Accepted	0.16 sec, 34.01 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Anagram

Sebuah anagram adalah kata yang bisa dibentuk dengan mengubah urutan huruf-huruf pada sebuah kata. Sebagai contoh, kata "kasur" bisa diubah urutannya menjadi "rusak"`, kata "bisa" bisa diubah urutannya menjadi "sabi".

Dari contoh tersebut, bisa disimpulkan bahwa kata "kasur" merupakan anagram dari "rusak" dan kata "bisa" adalah anagram dari "sabi".

Pada soal kali ini, buatlah sebuah program Main.java yang memenuhi spesifikasi berikut:
- Program menerima input berikut:
  - Pada baris pertama, berisi bilangan bulat N (1 ≤ N ≤ 100) menyatakan banyak kata yang akan diterima program.
  - Baris berikutnya berisi N buah kata yang dipisahkan spasi. Masing-masing kata hanya mengandung huruf kecil alfabet dan dijamin memiliki panjang ≤ 50.
- Program mengeluarkan output berupa banyaknya anagram unik dari seluruh kata yang diterima.

Contoh Input

```
4
kasur bisa sabi rusak
```

Contoh Output

```
2
```

Penjelasan

Himpunan {"kasur". "rusak"} memiliki anagram yang sama, begitu juga dengan himpunan {"bisa", "sabi"}.
Sehingga dari seluruh kata yang diterima, didapat 2 anagram unik.

Submit file **Main.java.**

Hints:
Manfaatkan pengurutan *sorting* terhadap karakter pada kata untuk menentukan apakah kata satu dengan yang lainnya memiliki anagram yang sama.

Java 8

Main.java

Score: 100

Blackbox
Score: 100
Verdict: Accepted
Evaluator: Exact

No	Score	Verdict	Description
1	5	Accepted	0.27 sec, 28.39 MB
2	5	Accepted	0.08 sec, 28.52 MB
3	5	Accepted	0.06 sec, 28.66 MB
4	5	Accepted	0.06 sec, 28.98 MB
5	5	Accepted	0.07 sec, 28.54 MB
6	5	Accepted	0.06 sec, 28.16 MB
7	5	Accepted	0.07 sec, 28.99 MB
8	5	Accepted	0.06 sec, 28.82 MB
9	5	Accepted	0.07 sec, 28.01 MB
10	5	Accepted	0.17 sec, 29.12 MB
11	10	Accepted	0.16 sec, 29.42 MB
12	10	Accepted	0.18 sec, 28.61 MB
13	10	Accepted	0.14 sec, 28.09 MB
14	10	Accepted	0.11 sec, 29.00 MB
15	10	Accepted	0.09 sec, 29.50 MB

◄ Praktikum 8

Jump to...