

Started on	Thursday, 18 May 2023, 6:16 PM
State	Finished
Completed on	Thursday, 18 May 2023, 6:53 PM
Time taken	37 mins
Grade	600.00 out of 600.00 (100%)

Question **1**

Correct

Mark 100.00 out of 100.00

Time limit	1.5 s
Memory limit	64 MB

Deque atau **Double Ended Queue** adalah versi umum dari struktur data **Queue** yang memungkinkan penyisipan dan penghapusan di kedua ujungnya.

Buatlah sebuah program **Main.java** (Buatlah sendiri) yang menggunakan **Deque** dari **Collection Java** untuk memenuhi spesifikasi program sebagai berikut:

- Program menerima input sebagai berikut:
 - Pada baris pertama, berisi bilangan bulat positif **Q** ($1 \leq Q \leq 100000$) menyatakan banyaknya operasi yang akan dilakukan
 - **Q** baris berikutnya berisi salah satu dari operasi berikut
 - **addFirst X** → Masukkan bilangan bulat **X** kedalam **Deque** dari depan
 - **addLast X** → Masukkan bilangan bulat **X** kedalam **Deque** dari belakang
 - **pollFirst** → Mengeluarkan nilai bilangan di indeks pertama **Deque** ke layar, kemudian menghapus bilangan tersebut
 - **pollLast** → Mengeluarkan nilai bilangan di indeks terakhir **Deque** ke layar, kemudian menghapus bilangan tersebut
 - **reverse** → Melakukan reverse semua elemen pada **Deque**
- Program mengeluarkan output sebagai berikut
 - Untuk setiap operasi **pollFirst** dan **pollLast** keluarkan bilangan yang sesuai
 - apabila pada operasi tersebut **Deque** sedang kosong, maka keluarkan **“EMPTY”** (tanpa tanda petik) pada layar

Contoh Input

```
15
addFirst 93
pollFirst
pollLast
reverse
pollLast
reverse
addFirst 80
addLast 53
addLast 50
pollFirst
pollFirst
reverse
addLast 66
reverse
pollFirst
```

Contoh Output

```
93
EMPTY
EMPTY
80
53
66
```

Submit file **Main.java**.

Hints:

- Anda dapat menggunakan **Deque** sebagai berikut, `Deque<Integer> deque = new ArrayDeque<Integer>()`
- Untuk melakukan operasi yang dibutuhkan, gunakan *method* yang ada pada format input (kecuali **reverse** yang bukan merupakan *method* **Deque**)

Java 8

 [Main.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.08 sec, 27.99 MB
2	10	Accepted	0.08 sec, 28.60 MB
3	10	Accepted	0.08 sec, 28.97 MB
4	10	Accepted	0.09 sec, 28.91 MB
5	10	Accepted	0.08 sec, 27.88 MB
6	10	Accepted	0.08 sec, 27.85 MB
7	10	Accepted	0.09 sec, 29.83 MB
8	10	Accepted	0.69 sec, 65.00 MB
9	10	Accepted	1.46 sec, 53.47 MB
10	10	Accepted	0.78 sec, 56.57 MB

Question **2**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Buatlah sebuah program Java yang menerima barisan bilangan bulat yang dipisahkan oleh spasi kemudian mengembalikan list baru yang terurut secara menaik tanpa ada nilai yang duplikat. Jangan lupa new line pada akhir baris

Aturan masukan yang diterima:

- List input dapat mengandung bilangan negatif
- List input dapat berupa list kosong
- List input dapat mengandung hingga 1000 bilangan

Asumsi:

- List input pasti berupa list bilangan bulat yang dipisahkan oleh spasi (contoh: 5 3 9 1 3 8 1 2 2)

Contoh:

Input: 5 3 9 1 3 8 1 2 2

Output: [1, 2, 3, 5, 8, 9]

Silahkan lengkapi file [Main.java](#)

Hint:

- Gunakan konsep collection List dan Set
- Gunakan fungsi .split untuk memisahkan elemen dalam string
- Format output adalah format print ArrayList. Silahkan langsung print variabel bertipe ArrayList

Java 8

 [Main.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	12	Accepted	0.07 sec, 28.92 MB
2	12	Accepted	0.07 sec, 30.68 MB
3	12	Accepted	0.07 sec, 28.35 MB
4	12	Accepted	0.07 sec, 26.61 MB
5	12	Accepted	0.07 sec, 28.41 MB
6	12	Accepted	0.07 sec, 28.88 MB
7	12	Accepted	0.07 sec, 28.08 MB
8	16	Accepted	0.07 sec, 28.04 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Student Sorter

Buatlah sebuah program Java yang mengelola informasi mengenai siswa dan nilai-nilai mereka dalam beberapa mata pelajaran. Program ini akan menghitung nilai rata-rata (GPA) dari setiap siswa dan mengurutkan mereka berdasarkan GPA.

Program harus memenuhi persyaratan berikut:

Dalam kelas StudentSorter:

- Terdapat kelas Student dengan atribut name (String) dan courseGrades (Map<String, Integer>). Kelas Student memiliki method **addCourseGrade** untuk menambahkan nilai mata pelajaran dan **getGPA** menghitung GPA.
- Terdapat kelas StudentComparator yang mengimplementasikan Comparator. Kelas ini harus mengurutkan objek Student berdasarkan GPA. Kelas ini meng-override method **compare** yang melakukan komparasi siswa s1 dan s2. Cara kerja komparasi adalah jika s1 < s2 method ini akan mengembalikan bilangan negatif, nol jika sama dan positif jika s1 > s2. Bilangan tersebut bebas.
- Terdapat static method **sortStudentsByGPA** yang mengembalikan list objek Student yang sudah diurutkan dengan StudentComparator

Lengkapilah file [StudentSorter.java](#).

Submit kembali file **StudentSorter.java** yang telah berisi jawaban Anda.

Java 8

 [StudentSorter.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	20	Accepted	0.07 sec, 28.37 MB
2	20	Accepted	0.08 sec, 27.92 MB
3	20	Accepted	0.07 sec, 28.17 MB
4	20	Accepted	0.07 sec, 28.00 MB
5	20	Accepted	0.07 sec, 28.30 MB

Question **4**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Set adalah sebuah struktur data yang memiliki kemampuan untuk menyimpan elemen-elemen unik. **Set** merupakan implementasi himpunan terbatas pada matematika. **Set** sebagai struktur data memiliki berbagai cara untuk diimplementasi, salah satu implementasi **Set** dalam bahasa Java adalah **TreeSet**.

TreeSet merupakan implementasi **Set** yang juga menyimpan informasi urutan dari elemen-elemen yang dimiliki. Karena urutan yang dipertahankan, pengambilan nilai minimal dan maksimal pada **TreeSet** memiliki kompleksitas waktu $O(1)$.

Buatlah sebuah program **Main.java** (Buatlah sendiri) yang memanfaatkan **TreeSet** dari **Collection Java** untuk memenuhi spesifikasi program sebagai berikut:

- Program menerima input sebagai berikut:
 - Pada baris pertama, berisi bilangan bulat positif **Q** ($1 \leq Q \leq 100$) menyatakan banyaknya operasi yang akan dilakukan
 - **Q** baris berikutnya berisi salah satu dari operasi berikut
 - **add X** → Masukkan bilangan bulat **X** kedalam **TreeSet**
 - **remove X**
 - Apabila terdapat bilangan bulat **X** pada **TreeSet**, maka hapus element tersebut
 - Apabila tidak ada, keluarkan pada layar "**Element {X} is not in The TreeSet**" (Tanpa tanda petik)
 - **first**
 - Mengeluarkan nilai bilangan minimal dari **TreeSet** ke layar
 - Apabila **TreeSet** kosong, keluarkan pada layar "**EMPTY**" (Tanpa tanda petik)
 - **last**
 - Mengeluarkan nilai bilangan maksimal dari **TreeSet** ke layar
 - Apabila **TreeSet** kosong, keluarkan pada layar "**EMPTY**" (Tanpa tanda petik)
- Program mengeluarkan output sebagai berikut
 - Untuk setiap operasi **remove**, **first**, dan **last** keluarkan sesuai yang diharapkan.

Contoh Input

```
8
add 10
remove 9
remove 10
add 5
add 7
add 8
first
last
```

Contoh Output

```
Element 9 is not in The TreeSet
5
8
```

Submit file **Main.java**.

Hints:

- Anda dapat menggunakan **TreeSet** sebagai berikut, **TreeSet<Integer> TreeSet = new TreeSet<Integer>()**
- Untuk melakukan operasi yang dibutuhkan, gunakan *method* yang ada pada format input.
- Untuk melihat apakah terdapat element **x** pada **TreeSet** gunakan *method* **contains(X)** pada **TreeSet**.

Java 8

 [Main.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.08 sec, 28.06 MB
2	10	Accepted	0.08 sec, 28.40 MB
3	10	Accepted	0.08 sec, 28.80 MB
4	10	Accepted	0.08 sec, 28.89 MB
5	10	Accepted	0.10 sec, 28.26 MB
6	10	Accepted	0.08 sec, 28.93 MB
7	10	Accepted	0.08 sec, 27.91 MB
8	10	Accepted	0.08 sec, 28.45 MB
9	10	Accepted	0.08 sec, 28.13 MB
10	10	Accepted	0.09 sec, 28.90 MB

Question **5**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Buatlah sebuah program Java yang menerima dua baris **string** yang dipisahkan oleh spasi kemudian mengembalikan satu list baru yang merupakan elemen yang sama yang dimiliki oleh kedua baris string tersebut

Aturan masukan yang diterima:

- List input dapat berupa list kosong
- List input dapat mengandung kata yang duplikat

Asumsi:

- List input pasti berupa list string yang dipisahkan oleh spasi (contoh: saya senang praktikum oop)

Contoh:

Input:

apple banana cherry date elderberry

date elderberry fig grape honeydew

Output: [date, elderberry]

Silahkan lengkapi file [Main.java](#)

Hint:

- Gunakan konsep collection List dan Set
- Gunakan fungsi .split untuk memisahkan elemen dalam string
- Format output adalah format print ArrayList. Silahkan langsung print variabel bertipe ArrayList.
- Jika terdapat lebih dari satu elemen dengan nilai yang sama dari kedua string, maka cukup ambil salah satu saja
- Urutan output terurut berdasarkan elemen pertama yang memiliki nilai yang sama pada kedua baris string

Java 8

 [Main.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	12	Accepted	0.13 sec, 29.75 MB
2	12	Accepted	0.12 sec, 28.36 MB
3	12	Accepted	0.11 sec, 28.94 MB
4	12	Accepted	0.10 sec, 28.38 MB
5	12	Accepted	0.09 sec, 27.77 MB
6	12	Accepted	0.07 sec, 28.33 MB
7	12	Accepted	0.07 sec, 28.62 MB
8	16	Accepted	0.06 sec, 28.00 MB

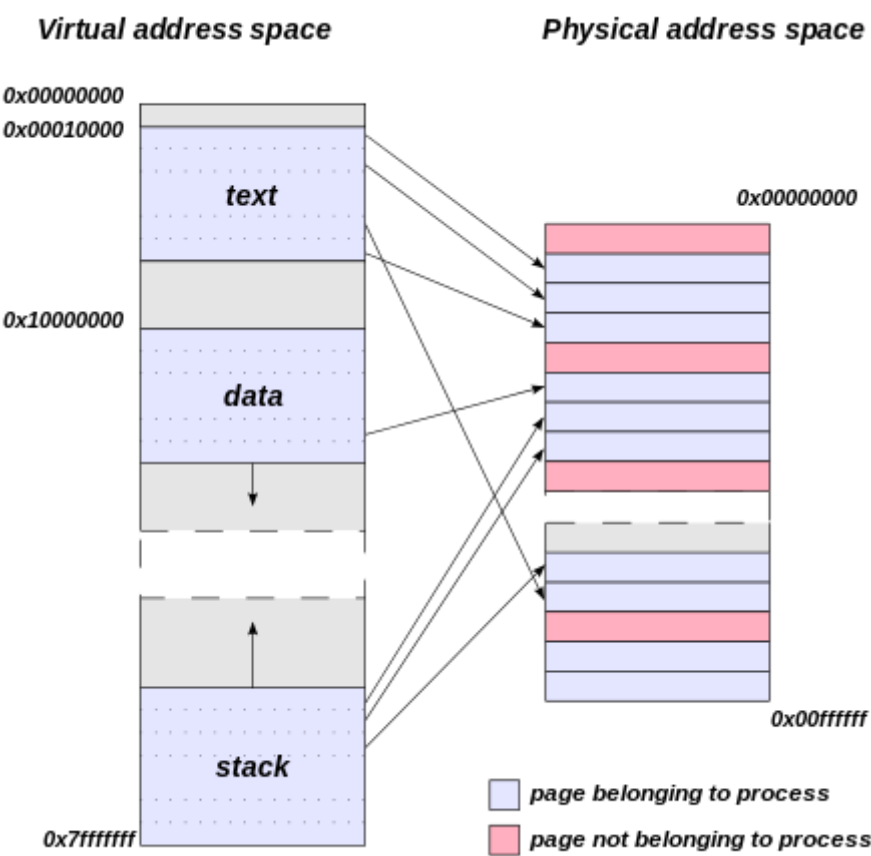
Time limit	1 s
Memory limit	64 MB

Page Replacement LRU

Latar Belakang

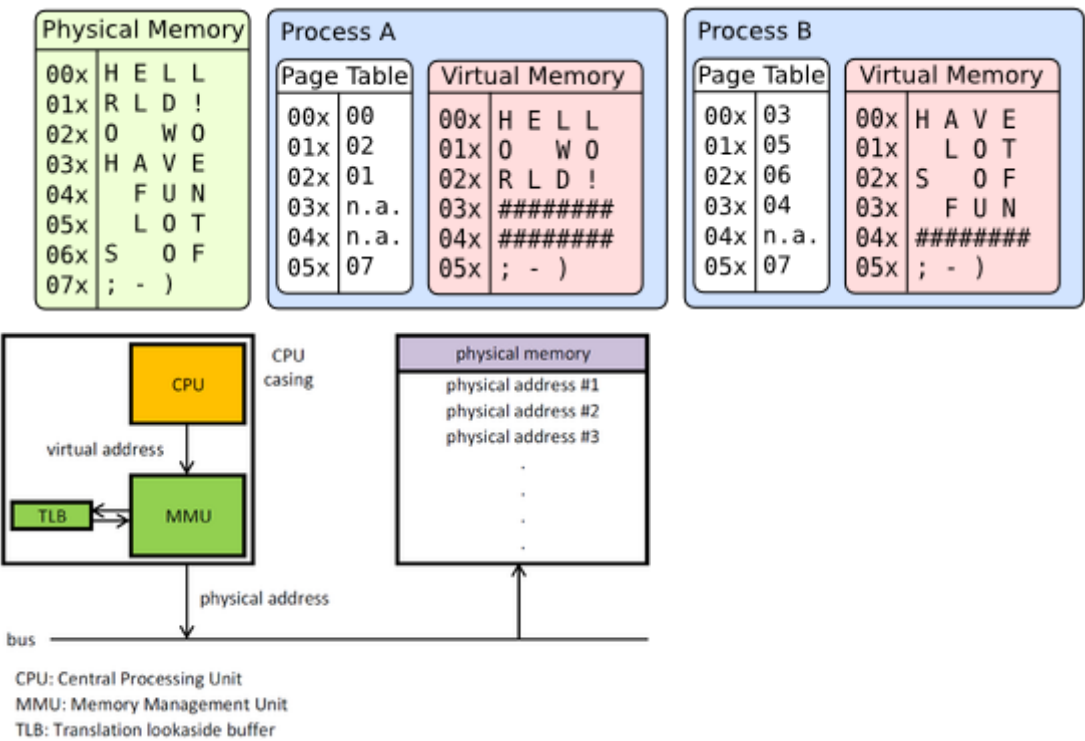
"Process" merupakan suatu program yang sedang berjalan. Pada OS Linux, setiap process mendapatkan "virtual memory address space" tersendiri. Maksudnya bagaimana? Misalnya pada PC dengan memory/RAM 4GB memiliki process Chrome dan VScode, dari kacamata masing-masing process tersebut, masing-masing "merasa" memiliki akses penuh RAM 4GB atau bisa lebih! Tetapi kenyataannya, hanya terdapat akses terbatas pada RAM. OS dan hardware bersama-sama membangun ilusi tersebut dengan beberapa mekanisme seperti mapping virtual address ke physical address, paging, swapping, etc.

"virtual memory" suatu process secara kasarnya akan dipotong-potong, potongan tersebut bernama "page" yang kemudian ditempatkan pada physical memory. Mekanisme tersebut dikenal dengan "paging", berikut ilustrasi paging suatu process



(Sumber: [Wikipedia](#))

Disini tugas OS pada dasarnya melakukan "tracking" page-page untuk setiap process pada "page table", sehingga ketika ada permintaan untuk akses suatu alamat, MMU dapat melakukan mapping address dengan benar



(Sumber: [OSdev](#), [Wikipedia](#))

Seperti yang dikatakan sebelumnya, setiap proses memiliki virtual memori sendiri sehingga ada kemungkinan physical memori **tidak dapat menampung semua page processes**, saat kondisi ini terjadi OS umumnya perlu mekanisme/algoritma "**Page Replacement**" contohnya **LRU** agar page lain dapat masuk ke physical memori dengan me-replace page yang jarang diakses, page tersebut kemudian dipindahkan ke disk. Skema ini sering disebut dengan

"swapping". Kasus lain yang perlu diperhatikan adalah ketika suatu page ingin diakses tetapi page tersebut tidak ada di physical memori sehingga akan terjadi "page fault" . OS kemudian perlu memuat page tersebut dari disk ke memori lagi serta memperbarui "page table" process tersebut.

Problem Statement

Tugas Anda disini adalah mengimplementasikan algoritma "Page Replacement" LRU (Least Recently Used) dengan menggunakan HashMap dan Doubly linked list. HashMap bertugas untuk penyimpanan page dan linked list untuk tracking posisi page.

Detil Implementasi:

- Saat adanya penambahan page baru (addPage) dan ukuran HashMap/LinkedList = maxPages, page paling belakang pada LinkedList akan dihapus dan page baru ditempatkan di depan LinkedList. Jika ukurannya masih kurang dari maxPages page baru tetap ditempatkan paling depan tanpa adanya penghapusan
- Saat adanya pengaksesan suatu page (accessPage) dan page tersebut tidak ada di LinkedList, akan dikeluarkan pesan "page fault!" ke stdout lalu accessPage mengembalikan -1. Jika page yang diakses ada di LinkedList, ambil page tersebut di HashMap, pindahkan page tersebut ke depan LinkedList dan kembalikan physicalAddress

Agar lebih jelas, langsung saja dengan contoh:

Contoh kasus:

```
Misalnya maxPages = 3
// accessPage(<PID, pageID>, physicalAddress)
addPage(<3, 5>, 1)
addPage(<1, 2>, 4)
accessPage(<1, 2>)
addPage(<1, 3>, 7)
addPage(<2, 5>, 7)
accessPage(<3, 5>, 1)
accessPage(<2, 5>, 7)
```

Jika setiap kembalian dari fungsi accessPage di-outputkan:

```
4
page fault!
-1
7
```

State LinkedList setiap tahap akses:

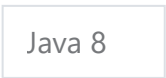
```
saat accessPage(<1, 2>) :
(1,2) <-> (3,5) <-> null
saat accessPage(<3, 5>):
(2, 5) <-> (1, 3) <-> (1, 2)
karena page (3, 5) sudah tidak ada di memori maka akan dikeluarkan "page fault!" dan kembalian -1
saat accessPage(<2, 5>):
(2, 5) <-> (1, 3) <-> (1, 2)
```

Files

Lengkapilah file [PageReplacementLRU.java](#), disediakan juga [Pair.java](#) dan [PageReplacementLRUDriver.java](#) untuk menjalankan dan mengecek kebenaran awal program Anda

Submit kembali file **PageReplacementLRU.java** yang telah berisi jawaban Anda.

Selamat bekerja :)



[PageReplacementLRU.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	20	Accepted	0.40 sec, 27.03 MB
2	20	Accepted	0.32 sec, 26.75 MB
3	20	Accepted	0.23 sec, 28.93 MB
4	20	Accepted	0.25 sec, 28.40 MB
5	20	Accepted	0.14 sec, 28.50 MB

[◀ Feedback Praktikum 9](#)

Jump to...