UNIVERSITI MALAYA
UNIVERSITY OF MALAYA

PEPERIKSAAN IJAZAH SARJANA MUDA SAINS KOMPUTER
EXAMINATION FOR THE DEGREE OF BACHELOR OF COMPUTER SCIENCE

SESI AKADEMIK 2019/2020 : SEMESTER I
ACADEMIC SESSION 2019/2020 : *SEMESTER I*

WIA1002 : Struktur Data
Data Structure

Dis 2019/Jan 2020 Masa: 3 jam 30 minit
Dec 2019/Jan 2020 *Time: 3 hours 30 minutes*

ARAHAN KEPADA CALON:
INSTRUCTIONS TO CANDIDATES:

Calon dikehendaki menjawab **SEMUA** soalan (50 markah).
*Answer **ALL** questions (50 marks).*

Satu direktori akaun peperiksaan yang disediakan untuk diakses oleh setiap calon. Untuk setiap soalan peperiksaan, sediakan satu fail java untuk disimpan dan disalinkan ke dalam direktori akaun peperiksaan tersebut. Calon tidak perlu menghantar apa-apa pakej java. Direktori akaun peperiksaan calon hanya perlu mempunyai fail-fail seperti berikut:

An exam account directory will be provides and is accessible to each candidate. For each exam question, prepare a java file to be saved and copied into the exam account directory. Candidate is not required to submit any java packages. The candidate's exam account directory should only contain the following files:

MatrixNum_Q1.java
MatrixNum_Q2.java
MatrixNum_Q3.java
MatrixNum_Q4.java

(Kertas soalan ini mengandungi 4 soalan dalam 12 halaman bercetak)
(This question paper consists of 4 questions on 12 printed pages)

1. Salah satu teknik popular struktur data adalah pelaksanaan timbunan generik. Tuliskan suatu program menggunakan timbunan generik. Program anda mesti mempunyai metod-metod berikut (lihat Jadual 1).

One popular technique in data structure is the generic stack implementation. Write a program to implement generic stack. Your program must contain the following methods (see Table 1).

Jadual 1: Senarai nama-nama metod dan spesifikasinya
Table 1: List of method names and their specification

Pembina/Nama Metod *Constructor/Method name*	Spesifikasi *Specification*
i) Constructor for the generic stack class	Konstruktor lalai *Default constructor*
ii) isEmpty	Memulangkan sama ada timbunan generik tersebut adalah kosong atau tidak *Return whether or not the generic stack is empty*
iii) isFull	Memulangkan sama ada timbunan generik tersebut adalah penuh atau tidak *Return whether or not the generic stack is full*
iv) peek	Memulangkan elemen teratas dalam timbunan generik tanpa membuangnya. *Return the value of the top element in the generic stack without removing it*
v) push	Menambah elemen ke posisi teratas dalam timbunan generik *Add element to the top of generic stack*
vi) pushMany	Menambah beberapa elemen ke dalam timbunan generik menggunakan koma sebagai pembatas *Add several elements to the generic stack using comma as split delimiter*
vii) pop	Membuang elemen di posisi teratas dalam timbunan generik *Remove element from the top of the generic stack*
viii) popMiddle	Membuang elemen di posisi tengah jika bilangan elemen dalam timbunan generik adalah ganjil *Remove element from the middle position if the number of element in the generic stack is odd number*
ix) popAll	Membuang kesemua elemen-elemen di dalam timbunan generik *Remove all elements from the generic stack*
x) display	Memaparkan kesemua elemen-elemen di dalam timbunan generik *Display all elements in the generic stack*

Tulis kesemua metod-metod dalam Jadual 1 serta metod *main()* anda di dalam fail program yang sama. Metod *main()* anda perlu mengikuti cadangan seperti dalam Rajah 1(a). Ubahsuai dan uji metod-metod anda supaya menyerupai output di dalam Rajah 1(b).

Write all methods from Table 1 as well as your main() method in the same program file. Your main() method should follow the suggestion shown in Figure 1(a). Modify and test your methods so you get the same output shown in Figure 1(b).

```
public static void main(String args[]){

    GenericStack<String> stack1 = new GenericStack<String>(7);
    stack1.push("apple");
    stack1.display();
    stack1.pushMany("broccoli,chicken sandwich,donut,french
fries,juice,maruku");
    stack1.display();
    System.out.println("Pop the top of the stack: " + stack1.pop());
    System.out.println("Pop the top of the stack: " + stack1.pop());
    stack1.display();
    System.out.println("Pop middle of the stack: " + stack1.popMiddle());

    stack1.display();
    System.out.println("Pop middle of the stack: " + stack1.popMiddle());
    System.out.println("----------");
    GenericStack<Integer> stack2 = new GenericStack<Integer>(10);
    stack2.push(1);
    stack2.push(2);
    stack2.pushMany("3 4,5,6 7");
    stack2.display();
    stack2.popAll();
    stack2.display();
  }
```

Rajah 1(a): Metod *main()*
Figure 1(a): Main() method

```
run:
Push: apple

There are 1 items in the stack. Displaying...
apple

Push many into stack....
Push: broccoli
Push: chicken sandwich
Push: donut
Push: french fries
Push: juice
Push: maruku

There are 7 items in the stack. Displaying...
maruku
juice
french fries
donut
```

```
chicken sandwich
broccoli
apple

Pop the top of the stack: maruku
Pop the top of the stack: juice

There are 5 items in the stack. Displaying...
french fries
donut
chicken sandwich
broccoli
apple

Pop middle of the stack: chicken sandwich

There are 4 items in the stack. Displaying...
french fries
donut
broccoli
apple

Current count of stack is even number, so no middle index..
Pop middle of the stack: null

There are 4 items in the stack. Displaying...
french fries
donut
broccoli
apple

-----------------------------------------------
Push: 1
Push: 2
Push many into stack....
Push: 3 4
Push: 5
Push: 6 7

There are 5 items in the stack. Displaying...
6 7
5
3 4
2
1

There are 5 items in the stack. Pop all...
Removing 6 7 ..
Removing 5 ..
Removing 3 4 ..
Removing 2 ..
Removing 1 ..

Stack is empty, nothing to display...


BUILD SUCCESSFUL (total time: 0 seconds)
```

Rajah 1(b): Output

Figure 1(b): Output

* Markah tidak akan diberikan bagi apa-apa pelaksanaan menggunakan kelas *Stack* atau mana-mana kelas *Collection* yang sedia ada didapati daripada *Java API Library.*

* *No marks will be given for any implementation using the existing Stack class or any other Collection classes available from the Java API Library.*

(15 markah/*marks*)

2. Baris-gilir menyokong operasi tambah dan operasi singkir berdasarkan polisi masuk dahulu keluar dahulu. Tuliskan suatu program menggunakan baris-gilir generik. Program anda mesti mempunyai metod-metod berikut (lihat Jadual 2).

Queue supports the insert and the remove operations based on the First-in First-out policy. Write a program to implement generic queue. Your program must contain the following methods (see Table 2).

Jadual 2: Senarai nama-nama metod dan spesifikasinya
Table 2: List of method names and their specification

Pembina/Nama Metod *Constructor/Method name*	Spesifikasi *Specification*
i) Constructor for the generic queue class	Konstruktor lalai *Default constructor*
ii) isEmpty	Memulangkan sama ada baris-gilir generik tersebut adalah kosong atau tidak *Return whether or not the generic queue is empty*
iii) isFull	Memulangkan sama ada baris-gilir generik tersebut adalah penuh atau tidak *Return whether or not the generic queue is full*
iv) peek	Memulangkan nilai bagi elemen pertama dalam baris-gilir generik tersebut *Return the value of the first element in the generic queue*
v) enqueue	Menambah elemen pada baris-gilir generik *Add element to the generic queue*
vi) dequeue	Membuang elemen dari baris-gilir generik *Remove element from the generic queue*
vii) changeOrder	Menerima satu parameter integer *k* yang mencapai satu elemen dalam baris-gilir generik dari hadapan. Kemudian, menukar susunan baris gilir generik dengan elemen di posisi k dianjak ke hadapan barisan. Tiada elemen disingkirkan dari barisan. *Accepts one integer parameter k which points to an element in the generic queue from front. Then, change the arrangement of the generics queue so element at position k becomes first in queue. No elements are removed from the queue.*

viii)display	Memaparkan elemen-elemen di dalam baris-gilir generik *Display all elements in the generic queue*

Tulis semua metod-metod dalam Jadual 2 serta metod *main()* anda di dalam fail program yang sama. Metod *main()* anda perlu mengikuti cadangan seperti dalam Rajah 2(a). Ubahsuai dan uji metod-metod anda supaya menyerupai output di dalam Rajah 2(b).

Write all methods from Table 2 as well as your main() method in the same program file. Your main() method should follow the suggestion shown in Figure 2(a). Modify and test your methods so you get the same output shown in Figure 2(b).

```
public static void main(String[] args) {

    GenericQueue<String> q = new GenericQueue<String>(10);
    q.enqueue("Dom T");
    q.enqueue("Rafa N");
    q.enqueueMany("Roger F,Daniil M,Novak D,Alex Z,Stef T,Karen K");
    q.display();
    q.dequeue();
    q.enqueue("Matt B");
    q.enqueue("Kei N");
    q.display();
    q.dequeueAll();
    q.display();
    System.out.println("----------------------------------------");

    GenericQueue<Integer> q2 = new GenericQueue<Integer>(10);
    q2.enqueue(10);
    q2.enqueue(20);
    q2.enqueueMany("30,40,50,60,70,80,90");
    q2.display();
    q2.changeOrder(5);
    q2.display();
}
```

Rajah 2(a): Metod *main()*
Figure 2(a): Main() method

```
run:
Enqueue: Dom T
Enqueue: Rafa N
Enqueue: Roger F
Enqueue: Daniil M
Enqueue: Novak D
Enqueue: Alex Z
Enqueue: Stef T
Enqueue: Karen K

There are 8 items in the queue. Displaying...
Dom T | Rafa N | Roger F | Daniil M | Novak D | Alex Z | Stef T | Karen K |
Dequeue: Dom T
Enqueue: Matt B
Enqueue: Kei N
```

```
There are 9 items in the queue. Displaying...
Rafa N | Roger F | Daniil M | Novak D | Alex Z | Stef T | Karen K | Matt B | Kei N |

There are 9 items in the queue. Removing them all ...
Dequeue: Rafa N
Dequeue: Roger F
Dequeue: Daniil M
Dequeue: Novak D
Dequeue: Alex Z
Dequeue: Stef T
Dequeue: Karen K
Dequeue: Matt B
Dequeue: Kei N

Nothing to display
------------------------------------

Enqueue: 10
Enqueue: 20
Enqueue: 30
Enqueue: 40
Enqueue: 50
Enqueue: 60
Enqueue: 70
Enqueue: 80
Enqueue: 90

There are 9 items in the queue. Displaying...
10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 |

Change queue order...

There are 9 items in the queue. Displaying...
50 | 60 | 70 | 80 | 90 | 10 | 20 | 30 | 40 |

BUILD SUCCESSFUL (total time: 0 seconds)
```

Rajah 2(b): Output
Figure 2(b): Output

* Markah tidak akan diberikan bagi apa-apa pelaksanaan menggunakan kelas *Queue* atau mana-mana kelas *Collection* yang sedia ada didapati daripada *Java API Library.*

* *No marks will be given for any implementation using the existing Queue class or any other Collection classes available from the Java API Library.*

(15 markah/*marks*)

3. Salah satu teknik popular struktur data adalah dipanggil senarai berpaut. Tuliskan suatu program menggunakan senarai berpaut. Program anda mesti mempunyai metod-metod berikut (lihat Jadual 3).

One popular technique in data structure is linked list. Write a program using linked list. Your program must contain the following methods (see Table 3).

Jadual 3: Senarai nama-nama metod dan spesifikasinya
Table 3: List of method names and their specification

Pembina/Nama Metod *Constructor/Method name*	Spesifikasi *Specification*
i) Constructor for the linked list class	Konstruktor lalai untuk kelas senarai berpaut *Default constructor for the linked list class*
ii) Constructor for the node class	Konstruktor lalai untuk kelas nod *Default constructor for the node class*
iii) isEmpty	Memulangkan sama ada senarai berpaut tersebut adalah kosong atau tidak *Return whether or not the linked list is empty*
iv) add	Menambah elemen ke dalam senarai berpaut *Add element to the linked list*
v) addAfter	Menambah elemen baru selepas elemen sedia ada di dalam senarai berpaut *Add new element after existing element in the linked list*
vi) remove	Membuang elemen tertentu dari senarai berpaut *Remove a particular element from the linked list*
vii) display	Memaparkan elemen-elemen di dalam senarai berpaut *Display all elements in the linked list*
viii) totalSold()	Mengira jumlah jualan yang dikumpul *Count the total sale collected*

Tulis semua metod-metod dalam Jadual 3 serta metod *main()* anda di dalam fail program yang sama. Metod *main()* anda perlu mengikuti cadangan seperti dalam Rajah 3(a). Ubahsuai dan uji metod-metod anda supaya menyerupai output di dalam Rajah 3(b).

Write all methods in Table 3 as well as your main() method in the same program file. Your main() method should follow the suggestion shown in Figure 3(a). Modify and test your methods so you get the same output shown in Figure 3(b).

```
public static void main(String[] args){

        LL myLL = new LL();

        myLL.createNode("Lord of The Rings", 500);
        myLL.createNode("Tale of the Body Thief", 1);
        myLL.createNode("Memnoch the Devil", 100);
        myLL.createNode("Heart of a Samurai", 10);
        myLL.display();
        myLL.addAfter("Memnoch the Devil", "White Crane", 10);
        myLL.addAfter("White Crane", "Memoirs of a Geisha", 90);
        myLL.removeNode("Harry Potter");
        myLL.removeNode("Heart of a Samurai");
        myLL.display();

}
```

Rajah 3(a): Metod *main()*
Figure 3(a): Main() method

```
run:

Displaying the Linked List **********************************************

Heart of a Samurai: 10,000,000 Sold
Memnoch the Devil: 100,000,000 Sold
Tale of the Body Thief: 1,000,000 Sold
Lord of The Rings: 500,000,000 Sold

Adding White Crane after Memnoch the Devil
Found Memnoch the Devil which is book number 2 in the linked list

Displaying the Linked List **********************************************

Heart of a Samurai: 10,000,000 Sold
Memnoch the Devil: 100,000,000 Sold
White Crane: 10,000,000 Sold
Tale of the Body Thief: 1,000,000 Sold
Lord of The Rings: 500,000,000 Sold

Adding Memoirs of a Geisha after White Crane
Found White Crane which is book number 3 in the linked list

Displaying the Linked List **********************************************

Heart of a Samurai: 10,000,000 Sold
Memnoch the Devil: 100,000,000 Sold
White Crane: 10,000,000 Sold
Memoirs of a Geisha: 90,000,000 Sold
Tale of the Body Thief: 1,000,000 Sold
Lord of The Rings: 500,000,000 Sold

Removing Harry Potter..
Couldn't find bookname..

Removing Heart of a Samurai..
Heart of a Samurai is the first book. Removing Heart of a Samurai from the linked
list
```

```
Displaying the Linked List **********************************************


Memnoch the Devil: 100,000,000 Sold
White Crane: 10,000,000 Sold
Memoirs of a Geisha: 90,000,000 Sold
Tale of the Body Thief: 1,000,000 Sold
Lord of The Rings: 500,000,000 Sold

BUILD SUCCESSFUL (total time: 0 seconds)
```

Rajah 3(b): Output
Figure 3(b): Output

* Markah tidak akan diberikan bagi apa-apa pelaksanaan menggunakan kelas *Linked List* atau mana-mana kelas *Collection* yang sedia ada didapati daripada *Java API Library*.

* *No marks will be given for any implementation using the existing Linked List class or any other Collection classes available from the Java API Library.*

(15 markah/*marks*)

4. Peta-pagar merupakan struktur yang menyimpan data dalam bentuk pasangan-pasangan Kekunci dan nilai. Tuliskan suatu program menggunakan peta-pagar. Program anda mesti mempunyai metod-metod berikut (lihat Jadual 4).

HashMap is a structure that stores data in the form of Key and value pairs. Write a program to implement HashMap. Your program must contain the following methods (see Table 4).

Jadual 4: Senarai nama-nama metod dan spesifikasinya
Table 4: List of method names and their specification

Pembina/Nama Metod *Constructor/Method name*	Spesifikasi *Specification*
i) Constructor for the HashMap class	Konstruktor lalai yang menerima parameter-parameter kekunci dan nilai pasangan *Default constructor which accepts Key and value pair parameters*
ii) get	Memulangkan *entry* yang dipetakan kepada kekunci di dalam peta-pagar *Return the entry mapped to key in the HashMap*
iii) put	Menambah *entry* baru sekiranya kekunci belum lagi dipetakan di dalam peta-pagar. Sebaliknya, mengemaskini *entry* lama sekiranya kekunci telah wujud di dalam peta-pagar. *Add new entry if Key is not yet mapped into the HashMap. Otherwise, update entry mapped to Key if the HashMap already contains the Key*

Tulis semua metod-metod dalam Jadual 4 serta metod *main()* anda di dalam fail program yang sama. Metod *main()* anda perlu mengikuti cadangan seperti dalam Rajah 4(a). Ubahsuai dan uji metod-metod anda supaya menyerupai output di dalam Rajah 4(b).

Write all methods in Table 4 as well as your main() method in the same program file. Your main() method should follow the suggestion shown in Figure 4(a). Modify and test your methods so you get the same output shown in Figure 4(b).

```
public static void main(String[] args) {

  MyHashMap myHashMap = new MyHashMap();

  System.out.println("New directory entry: ");
  myHashMap.put("BruceW", "011-8998990");
  myHashMap.put("DeanW", "017-2274000");
  myHashMap.put("TonyS", "019-4550800");
  myHashMap.put("LaraC", "014-6402009");
  System.out.println("");

  System.out.println("Get directory: ");
  Entry e1 = myHashMap.get("DeanW");
  System.out.println("DeanW: " + e1.getValue());
  Entry e2 = myHashMap.get("BruceW");
  System.out.println("BruceW: " + e2.getValue());
  System.out.println("");

  System.out.println("Update directory: ");
  myHashMap.put("BruceW", "011-5677900");
  myHashMap.put("JeanG", "019-9001123");

  // Get directory:
  Entry e3 = myHashMap.get("BruceW");
  System.out.println("BruceW: " + e3.getValue());
  Entry e4 = myHashMap.get("JeanG");
  System.out.println("JeanG: " + e4.getValue());

}
```

Rajah 4(a): Metod *main()*
Figure 4(a): Main() method

```
run:
New directory entry:

Get directory:
DeanW: 017-2274000
BruceW: 011-8998990

Update directory:
BruceW: 011-5677900
JeanG: 019-9001123

BUILD SUCCESSFUL (total time: 0 seconds)
```

Rajah 4(b): Output
Figure 4(b): Output

* Markah tidak akan diberikan bagi apa-apa pelaksanaan menggunakan kelas *HashMap* atau mana-mana kelas *Collection* yang sedia ada didapati daripada *Java API Library.*

* *No marks will be given for any implementation using the existing HashMap class or any other Collection classes available from the Java API Library.*

(5 markah/*marks*)

TAMAT
END