UNIVERSITI MALAYA
UNIVERSITY OF MALAYA

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

SESI AKADEMIK 2018/2019 : SEMESTER I
ACADEMIC SESSION 2018/2019 : *SEMESTER I*

WIA1002 : Struktur Data
Data Structure

Jan 2019
Jan 2019

Masa: 3 jam 30 minit
Time: 3 hours 30 minutes

ARAHAN KEPADA CALON :
INSTRUCTIONS TO CANDIDATES :

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

Calon dikehendaki menyediakan satu folder di dalam direktori akaun peperiksaan yang dinamakan semula dengan nombor matriks masing-masing. Untuk setiap soalan peperiksaan, sediakan satu fail java untuk disimpan dan disalinkan ke dalam folder tersebut. Calon tidak perlu menghantar apa-apa pakej java. Direktori akaun peperiksaan calon hanya perlu mempunyai folder dan fail-fail seperti berikut:

Candidate is required to prepare a folder in the exam account directory which is renamed according to own matrix numbers. For each exam question, prepare a java file to be saved and copied into said folder. Candidate is not required to submit any java packages. The candidate's exam account directory should only contain the following folder and files:

[WIX170012]
WIX170012_Q1.java
WIX170012_Q2.java
WIX170012_Q3.java
WIX170012_Q4.java

(Kertas soalan ini mengandungi 4 soalan dalam 10 halaman bercetak)
(This question paper consists of 4 questions on 10 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 adalan kosong atau tidak *Return whether or not the generic stack is empty*
iii) isFull	Memulangkan sama ada timbunan generik tersebut adalan 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 first 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) popAll	Membuang kesemua elemen-elemen di dalam timbunan generik *Remove all elements from the generic stack*
ix) display	Memaparkan kesemua elemen-elemen di dalam timbunan generik *Display all elements in the generic stack*

Tulis 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 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>(5);
    stack1.push("one");
    stack1.display();
    stack1.pushMany("two,three four,five,six seven");
    stack1.display();
    stack1.pop();
    stack1.pop();
    stack1.display();
    System.out.println("-------------------------");
    GenericStack<Integer> stack2 = new GenericStack<Integer>(5);
    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: one

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

Push: two
Push: three four
Push: five
Push: six seven

There are 5 items in the stack. Displaying...
six seven
five
three four
two
one

Pop the top of stack...
Pop the top of stack...

There are 3 items in the stack. Displaying...
three four
two
one

-------------------------
Push: 1
Push: 2
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. Removing 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.*

(14 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 adalan kosong atau tidak *Return whether or not the generic queue is empty*
iii) isFull	Memulangkan sama ada baris-gilir generik tersebut adalan 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) enqueueMany	Menambah beberapa elemen ke dalam baris-gilir generik menggunakan koma sebagai pembatas *Add several elements to the generic queue using comma as split delimiter*
vii) dequeue	Membuang elemen dari baris-gilir *Remove element from the generic queue*
viii) dequeueAll	Membuang kesemua elemen-elemen di dalam baris-gilir generik *Remove all elements from the generic queue*
ix) display	Memaparkan elemen-elemen di dalam baris-gilir generik *Display all elements in the generic queue*

Tulis 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 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>(6);
    q.enqueue("Hello");
    q.enqueueMany("all,who,wants,ice cream,or cookies");
    q.display();
    q.dequeue();
    q.dequeueAll();
    q.display();
    System.out.println("------------------------------");
    GenericQueue<Integer> q2 = new GenericQueue<Integer>(6);
    q2.enqueue(1000);
    q2.enqueue(2000);
    q2.enqueueMany("3000,4000 5000,6000,7000,8000 9000");
    q2.display();
    q2.dequeue();
    q2.dequeue();
    q2.enqueue(8000);
    q2.dequeueAll();
    q2.display();
}
```

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

```
run:
Enqueue: Hello
Enqueue: all
Enqueue: who
Enqueue: wants
Enqueue: ice cream
Enqueue: or cookies

There are 6 items in the queue. Displaying...
Hello
all
who
wants
ice cream
or cookies

Dequeue: Hello

There are 5 items in the queue. Removing them all ...
Dequeue: all
Dequeue: who
Dequeue: wants
Dequeue: ice cream
Dequeue: or cookies

Nothing to display
------------------------------
Enqueue: 1000
Enqueue: 2000
Enqueue: 3000
Enqueue: 4000 5000
Enqueue: 6000
Enqueue: 7000
Queue is full

There are 6 items in the queue. Displaying...
1000
2000
3000
4000 5000
6000
7000

Dequeue: 1000
Dequeue: 2000
Enqueue: 8000

There are 5 items in the queue. Removing them all ...
Dequeue: 3000
Dequeue: 4000 5000
Dequeue: 6000
Dequeue: 7000
Dequeue: 8000

Nothing to display
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.*

(14 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) totalCreditHours()	Mengira jumlah jam kredit yang dikumpul *Count the total credit hours collected*

Tulis 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 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){

    LinkedList list = new LinkedList();

    list.add("Computing Mathematics", 3);
    list.add("Network Architecture", 3);
    list.add("Final Year Project", 5);
    list.add("Data Structure", 5);
    list.display();
    list.totalCreditHours();
    list.addAfter("Final Year Project", "Software Modelling", 4);
    list.addAfter("Software Modelling", "Intelligent Robot", 3);
    list.addAfter("Computing Mathematics", "Gamification", 4);
    list.totalCreditHours();
    list.remove("Network Architecture", 3);
    list.remove("Software Modelling", 4);
    list.display();
    list.totalCreditHours();
}
```

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

```
run:

Displaying the Linked List ----------------------

Data Structure: 5 hours
Final Year Project: 5 hours
Network Architecture: 3 hours
Computing Mathematics: 3 hours

Total credit hours: 16

Adding Software Modelling after Final Year Project

Adding Intelligent Robot after Software Modelling

Adding Gamification after Computing Mathematics

Total credit hours: 27

Removing Network Architecture..

Removing Software Modelling..

Displaying the Linked List ----------------------

Data Structure: 5 hours
Final Year Project: 5 hours
Intelligent Robot: 3 hours
Computing Mathematics: 3 hours
Gamification: 4 hours

Total credit hours: 20
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.*

(14 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 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 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();

    myHashMap.put("BruceW", "011-8998990");
    myHashMap.put("DeanW", "017-2274000");
    myHashMap.put("TonyS", "019-4550800");
    myHashMap.put("LaraC", "014-6402009");

    Entry e1 = myHashMap.get("DeanW");
    System.out.println("Value: " + e1.getValue());
    Entry e2 = myHashMap.get("TonyS");
    System.out.println("Value: " + e2.getValue());

    myHashMap.put("BruceW", "011-5677900");
    Entry e3 = myHashMap.get("BruceW");
    System.out.println("Value: " + e3.getValue());

    myHashMap.put("JeanG", "019-9001123");
    Entry e4 = myHashMap.get("JeanG");
    System.out.println("Value: " + e4.getValue());
}
```

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

```
run:
Value: 017-2274000
Value: 019-4550800
Value: 011-5677900
Value: 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.*

(8 markah/*marks*)

TAMAT
END