

UNIVERSITI MALAYA
UNIVERSITY OF MALAYA

PEPERIKSAAN IJAZAH SARJANA MUDA TEKNOLOGI MAKLUMAT
EXAMINATION FOR THE DEGREE OF BACHELOR OF INFORMATION TECHNOLOGY

SESI AKADEMIK 2017/2018 : SEMESTER I
ACADEMIC SESSION 2017/2018 : SEMESTER I

WIA1002 : Struktur Data
Data Structure

Jan 2018
Jan 2018

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).



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

1. Masalah *Towers of Hanoi* adalah satu masalah klasik yang sering digunakan untuk menggambarkan kekuatan teknik rekursi. Penjelasan masalah tersebut adalah seperti berikut:

The Towers of Hanoi problem is a classical problem used to illustrate the power of recursion. Explanation of the problem goes as follows:

Terdapat tiga tiang dengan 64 cakera yang berlainan saiz. Asalnya, semua cakera-cakera tersebut telah diletakkan pada tiang pertama dengan cakera terbesar berada paling bawah dan yang terkecil berada paling atas. Anda perlu menggerakkan kesemua cakera-cakera yang terletak di tiang pertama ke tiang ketiga, dengan cakera terbesar berada paling bawah dan yang terkecil berada paling atas. Anda hanya boleh menggerakkan hanya satu cakera pada satu-satu masa dan cakera yang lebih besar tidak boleh diletakkan di atas cakera yang lebih kecil. Tiang kedua boleh digunakan sebagai tiang pengantara untuk membantu anda memindahkan cakera-cakera tersebut.

There are three poles and 64 discs of different sizes. Initially, all the discs are placed on the first pole with the largest disc at the bottom and the smallest one at the top. You need to move all the discs from the first pole to the third pole, with the smallest disc at the top and the largest at the bottom. You can move only one disc at a time and at any point of time, a larger disc cannot be placed over a smaller one. The second pole can be used as an intermediate pole to help you in transferring the discs.



Bagi masalah dengan hanya 2 cakera (cakera 1 dan cakera 2), masalah ini boleh diselesaikan dengan langkah-langkah (algorithm) berikut:

- Langkah 1: Gerak cakera 1 dari tiang 1 ke tiang 2
- Langkah 2: Gerak cakera 2 dari tiang 1 ke tiang 3
- Langkah 3: Gerak cakera 1 dari tiang 2 ke tiang 3

For a puzzle with 2 discs (disc 1 and disc 2), the puzzle can be solved with the following steps (algorithm):

- Step 1: Move disc 1 from pole 1 to pole 2*
- Step 2: Move disc 2 from pole 1 to pole 3*
- Step 3: Move disc 1 from pole 2 to pole 3*

Tuliskan sebuah program untuk menyelesaikan masalah *Towers of Hanoi* dengan menggunakan teknik rekursi.

(Simpan dan salin fail ini ke direktori akaun peperiksaan anda dan masukkan ke dalam folder yang dinamakan [Q1]).

Write a program to solve Towers of Hanoi using recursion.

(Save and copy this file to your exam account directory and keep it inside a folder called [Q1]).

(7 markah/marks)

2. Baris-gilir menyokong operasi tambah dan operasi singkir berdasarkan polisi masuk dahulu keluar dahulu. Tulis sebuah program untuk membina sebuah kelas yang bernama *Queue*. Kelas ini seharusnya mempunyai metod-metod seperti yang dinyatakan dalam Jadual 1:

Queue supports the insert and the remove operations based on the First-in First-out policy. Write a program to create a class named Queue. The class should consist of methods as specified in 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) <i>Constructor for Queue class</i>	Konstruktor lalai <i>Default constructor</i>
ii) <i>isEmpty</i>	Memulangkan sama ada baris-gilir tersebut adalah kosong atau tidak <i>Return whether or not the queue is empty</i>
iii) <i>peek</i>	Memulangkan nilai bagi elemen pertama dalam baris-gilir tersebut <i>Return the value of the first element in the queue</i>
iv) <i>enqueue</i>	Menambah elemen pada baris-gilir paling bawah <i>Add element to the bottom of the queue</i>
v) <i>dequeue</i>	Membuang elemen pada baris-gilir paling atas <i>Remove element from the top of the queue</i>
vi) <i>display</i>	Memaparkan elemen-elemen di dalam baris-gilir tersebut <i>Display all elements in the queue</i>

(8 markah/marks)

- a) Uji metod-metod tersebut dengan memprogram di dalam metod *main()* anda seperti contoh di dalam Rajah 1.

Test the methods by programming your main() method similar to the example in Figure 1.


```

public static void main(String[] args) {

    Queue q=new Queue(5);

    q.enqueue("hello");
    q.enqueue("all");
    q.enqueue("who");
    q.enqueue("wants");
    q.enqueue("ice cream");
    q.dequeue();
    q.dequeue();
    q.enqueue("1000");
    q.enqueue("2000");
    q.peek();
}

```

```

run:
Data hello inserted
Data all inserted
Data who inserted
Data wants inserted
Data ice cream inserted
Data hello has been removed
Data all has been removed
Data 1000 inserted
Data 2000 inserted
Peek(): who
BUILD SUCCESSFUL (total time: 0 seconds)

```

Rajah 1
Figure 1

(2 markah/marks)

- b) Ubahsuai metod *display()* untuk memaparkan setiap indeks dan elemen dalam gambarajah berkotak seperti contoh dalam Rajah 2.

Modify the method display() to display each index and element in boxes as shown in the example in Figure 2.

```

public static void main(String[] args) {

    Queue q=new Queue(5);

    q.enqueue("hello");
    q.enqueue("all");
    q.enqueue("who");
    q.enqueue("wants");
    q.enqueue("ice cream");
    q.dequeue();
    q.dequeue();
    q.enqueue("1000");
    q.enqueue("2000");
    q.peek();

    q.display();
}

```

```

run:
Data hello inserted
Data all inserted
Data who inserted
Data wants inserted
Data ice cream inserted
Data hello has been removed
Data all has been removed
Data 1000 inserted
Data 2000 inserted
Peek(): who

```

0	1	2	3	4
who	wants	ice cream	1000	2000

Rajah 2
Figure 2

(3 markah/marks)

(Simpan dan salin kesemua fail-fail yang berkaitan dengan program ini (kelas *Queue* dan metod *main()*) ke direktori akaun peperiksaan anda di dalam folder yang dinamakan [Q2]).

(Save and copy all related files to this program (*Queue* class and *main()* method) to your exam account directory and keep them inside a folder called [Q2]).

* 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.

3. Salah satu teknik popular struktur data adalah timbunan. Tuliskan suatu program menggunakan timbunan. Program anda mesti mempunyai metod-metod berikut (lihat Jadual 2).

One popular technique in data structure is stack. Write a program using stack. 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) <i>Constructor for Stack class</i>	Konstruktor lalai <i>Default constructor</i>
ii) <i>isEmpty</i>	Memulangkan sama ada timbunan tersebut adalah kosong atau tidak <i>Return whether or not the stack is empty</i>
iii) <i>peek</i>	Memulangkan elemen teratas dalam timbunan tanpa membuangnya. <i>Return the value of the first element in the stack without removing it</i>
iv) <i>push</i>	Menambah elemen di posisi teratas dalam timbunan <i>Add element to the top of stack</i>
v) <i>pop</i>	Membuang elemen di posisi teratas dalam timbunan <i>Remove element from the top of the stack</i>
vi) <i>popAll</i>	Membuang kesemua elemen-elemen di dalam timbunan <i>Remove all elements from the stack</i>
vii) <i>display</i>	Memaparkan elemen-elemen di dalam timbunan <i>Display all elements in the stack</i>
viii) <i>displayInReverse</i>	Memaparkan elemen-elemen di dalam timbunan mengikut susunan terbalik <i>Display all elements in the stack in reversed order</i>

(8 markah/marks)

- a) Uji metod-metod tersebut dengan memprogram di dalam metod *main()* anda seperti contoh di dalam Rajah 3.

Test the methods by programming your main() method similar to the example in Figure 3.

```

public static void main(String args[]){

    StackTest stack1 = new StackTest(4);
    stack1.push("one");
    stack1.push("two");
    stack1.push("three");
    stack1.push("four");
    stack1.displayReverse();
    stack1.popAll();
    stack1.displayReverse();

}

```

```

run:
Push item into stack: one
Push item into stack: two
Push item into stack: three
Push item into stack: four
Display stack in reverse:
four
three
two
one
Pop all item in stack:
Removing four ..
Removing three ..
Removing two ..
Removing one ..
Display stack in reverse:
null
null
null
null
BUILD SUCCESSFUL (total time: 0 seconds)

```

Rajah 3
Figure 3

(2 markah/marks)

- b) Dengan sedikit pengubahsuaian, tukarkan program timbunan anda dalam bentuk *Generic* supaya yang berikut dapat dilakukan (lihat Rajah 4).

With some modification, change your stack program into Generic form so that the following can be implemented (see Figure 4).


```

public static void main(String args[]){

    StackWithGeneric<String> stack1 = new StackWithGeneric<String>(4);
    stack1.push("one");
    stack1.push("two");
    stack1.push("three");
    stack1.push("four");
    System.out.println(stack1.peek());
    stack1.display();
    stack1.displayReverse();
    stack1.popAll();

    StackWithGeneric<Integer> stack2 = new StackWithGeneric<Integer>(4);
    stack2.push(100);
    stack2.push(200);
    stack2.push(300);
    stack2.push(400);
    System.out.println(stack2.peek());
    stack2.display();
    stack2.displayReverse();
    stack2.popAll();

}

```

run:	400
Push item into stack: one	Display stack:
Push item into stack: two	100
Push item into stack: three	200
Push item into stack: four	300
four	400
Display stack:	Display stack in reverse:
one	400
two	300
three	200
four	100
Display stack in reverse:	Pop all item in stack:
four	Removing 400 ..
three	Removing 300 ..
two	Removing 200 ..
one	Removing 100 ..
Pop all item in stack:	BUILD SUCCESSFUL (total time: 0 seconds)
Removing four ..	
Removing three ..	
Removing two ..	
Removing one ..	
Push item into stack: 100	
Push item into stack: 200	
Push item into stack: 300	
Push item into stack: 400	

Rajah 4
Figure 4

(5 markah/marks)

(Simpan dan salin kesemua fail-fail yang berkaitan dengan program ini (kelas *Stack* dan metod *main()*) ke direktori akaun peperiksaan anda di dalam folder yang dinamakan [Q3]).

(Save and copy all related files to this program (*Stack* class and *main()* method) to your exam account directory and keep them inside a folder called [Q3]).

* 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.

4. 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) <i>Constructor for the linked list class</i>	Konstruktor lalai untuk kelas senarai berpaut <i>Default constructor for the linked list class</i>
ii) <i>Constructor for the node class</i>	Konstruktor lalai untuk kelas nod <i>Default constructor for the node class</i>
iii) <i>isEmpty</i>	Memulangkan sama ada timbunan tersebut adalah kosong atau tidak <i>Return whether or not the stack is empty</i>
iv) <i>add</i>	Menambah elemen di posisi teratas dalam timbunan <i>Add element to the top of stack</i>
v) <i>addAfter</i>	Membuang elemen di posisi teratas dalam timbunan <i>Remove element from the top of the stack</i>
vi) <i>remove</i>	Membuang kesemua elemen-elemen di dalam timbunan <i>Remove all elements from the stack</i>
vii) <i>display</i>	Memaparkan elemen-elemen di dalam timbunan <i>Display all elements in the stack</i>

(10 markah/marks)

- a) Uji metod-metod tersebut dengan memprogram di dalam metod *main()* anda seperti contoh di dalam Rajah 5.

Test the methods by programming your main() method similar to the example in Figure 5.

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

    LList list = new LList();

    list.add("Computing Mathematics 1", 3);
    list.add("Principles of AI", 3);
    list.add("Programming 1", 5);
    list.add("Data Structure", 5);
    list.display();
    list.addAfter("Programming 1", "Software Architecture", 4);
    list.addAfter("Software Architecture", "Networking", 4);

    list.remove("Networking", 4);
    list.display();

    list.remove("Principles of AI", 3);
    list.display();

}
```

```
run:

Displaying the Linked List *****

Data Structure: 5 hours
Programming 1: 5 hours
Principles of AI: 3 hours
Computing Mathematics 1: 3 hours

Adding Software Architecture after Programming 1
Found Programming 1 which is book number 2 in the linked list

Displaying the Linked List *****

Data Structure: 5 hours
Programming 1: 5 hours
Software Architecture: 4 hours
Principles of AI: 3 hours
Computing Mathematics 1: 3 hours

Adding Networking after Software Architecture
Found Software Architecture which is book number 3 in the linked list
```

Rajah 5
Figure 5

(3 markah/marks)

- b) Tambah satu metod *totalCreditHours()* untuk mengira jumlah jam kredit yang dikumpul. Lihat contoh program dan output dalam Rajah 6.

Add a method *totalCreditHours()* to count the total credit hours collected. See Figure 6 for program and output example.

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

    LList list = new LList();

    list.add("Computing Mathematics 1", 3);
    list.add("Principles of AI", 3);
    list.add("Programming 1", 5);
    list.add("Data Structure", 5);
    //list.display();
    list.totalCreditHours();
    list.addAfter("Programming 1", "Software Architecture", 4);
    list.addAfter("Software Architecture", "Networking", 4);
    //list.display();
    list.totalCreditHours();
    list.remove("Networking", 4);
    //list.display();
    list.remove("Principles of AI", 3);
    //list.display();
    list.totalCreditHours();
}
```

```
run:
Total credit hours taken: 16

Adding Software Architecture after Programming 1
Found Programming 1 which is book number 2 in the linked list

Adding Networking after Software Architecture
Found Software Architecture which is book number 3 in the linked list
Total credit hours taken: 24

Removing Networking..
Found a match.. Networking is book number 4 in the linked list

Removing Principles of AI..
Found a match.. Principles of AI is book number 4 in the linked list
Total credit hours taken: 17
BUILD SUCCESSFUL (total time: 0 seconds)
```

Rajah 6
Figure 6

(2 markah/marks)

(Simpan dan salin kesemua fail-fail yang berkaitan dengan program ini (kelas *LList* dan metod *main()*) ke direktori akaun peperiksaan anda di dalam folder yang dinamakan [Q4]).

(Save and copy all related files to this program (LList class and main() method) to your exam account directory and keep them inside a folder called [Q4]).

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

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

TAMAT
END