UNIVERSITI MALAYA
UNIVERSITY OF MALAYA

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

SESI AKADEMIK 2016/2017 : SEMESTER I
ACADEMIC SESSION 2016/2017 : *SEMESTER I*

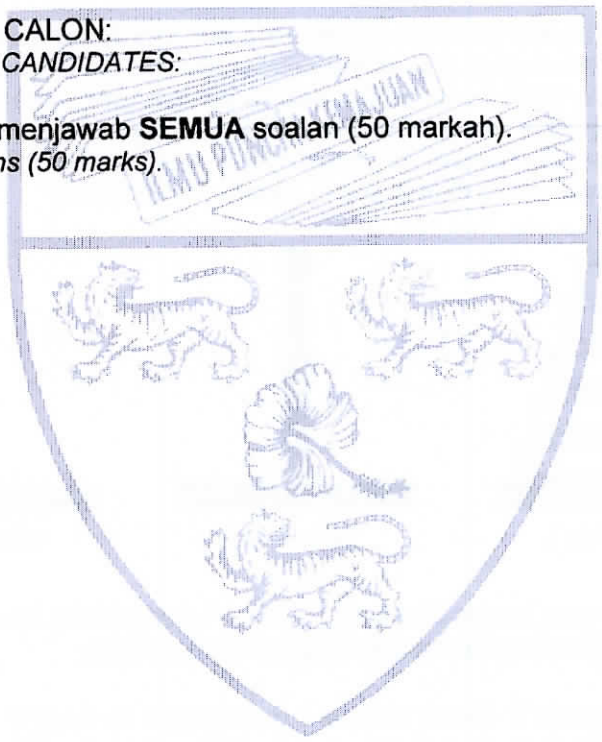
WIA1002 : Struktur Data
Data Structure

Dis 2016/Jan 2017
Dec 2016/Jan 2017

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).*



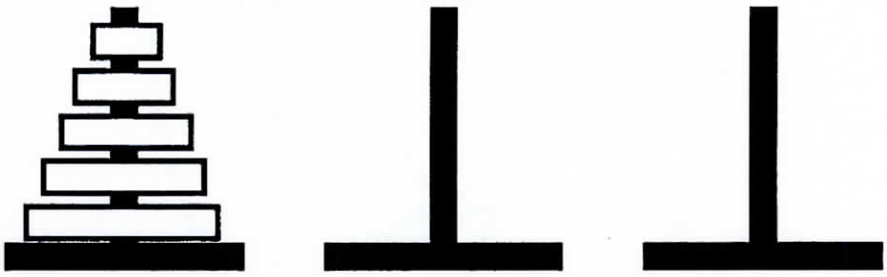
(Kertas soalan ini mengandungi 4 soalan dalam 8 halaman bercetak)
(This question paper consists of 4 questions on 8 printed pages)

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.



Tiang/*Pole* 1 Tiang/*Pole* 2 Tiang/*Pole* 3

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 1 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 3 from pole 1 to pole 3

a) Sekiranya anda diberi 3 cakera (cakera 1, cakera 2 dan cakera 3), rangka satu algoritma untuk menyelesaikan masalah *Towers of Hanoi*, di mana anda menggunakan teknik rekursi di dalam penyelesaian anda.

Assume that you are given 3 discs (disc 1, disc 2, disc and disc 3), devise an algorithm to solve the Towers of Hanoi problem, in which you are using recursion in your answers.

(3 markah/*marks*)

b) Berdasarkan algoritma yang anda tulis di (a), tuliskan sebuah program untuk menyelesaikan masalah *Towers of Hanoi* dengan menggunakan teknik rekursi.

Based on the algorithm that you wrote in (a), write a program to solve Towers of Hanoi using recursion.

(7 markah/*marks*)

2. Baris-gilir menyokong operasi tambah dan operasi singkir berdasarkan polisi masuk dahulu keluar dahulu.

Queue supports the insert and the remove operations based on the First-in First-out policy.

a) Tulis sebuah program untuk membina sebuah kelas yang bernama *Queue<Item>*. Kelas ini seharusnya mempunyai metod-metod seperti yang dinyatakan dalam Jadual 2:

Write a program to create a class named Queue<Item>. The class should consist of methods as specified in Table 2:

Pembina/Nama Metod *Constructor/Method name*	Spesifikasi *Specification*
i) Constructor for Queue class	Konstruktor lalai *Default constructor*
ii) isEmpty	Memulangkan sama ada baris-gilir tersebut adalan kosong atau tidak *Return whether or not the queue is empty*
iii) size	Memulangkan saiz baris-gilir tersebut *Return the size of the queue*
iv) peek	Memulangkan nilai bagi elemen pertama dalam baris-gilir tersebut *Return the value of the first element in the queue*
v) enqueue	Menambah elemen pada baris-gilir paling bawah *Add element to the bottom of the queue*
vi) dequeue	Membuang elemen pada baris-gilir paling atas *Remove element from the top of the queue*
vii) toString	Mencetak elemen-elemen di dalam baris-gilir

	tersebut *Print the element in the queue*

Jadual 1: Senarai Nama-Nama Metod dan Spesifikasinya
Table 1: List of method names and their specification

(11 markah/*marks*)

b) Bina sebuah metod bernama *ChangeOrder(int k)* yang menerima satu parameter integer *k*. Metod ini akan singkirkan *k-1* elemen daripada atas sesuatu barisan dan menambahkannya di bahagian bawah barisan yang sama. Elemen ke-*k* ini akan ditambah ke bawah barisan tetapi tidak akan disingkirkan daripada barisan. Sebagai contoh:

Create a method called ChangeOrder(int k) that accepts a parameter of integer k. This method will remove k-1 element from the top of a queue and add its corresponding value to the bottom of the same queue. The k-th element will be added at the bottom of the queue but will not be removed from the queue. For example:

Andaikan input / *Say input*: 10, 20, 30, 40, 50, 60, 70, 80, 90
Dan nilai / *And value of* k = 4,
Maka output / *Thus the output: 40, 50, 60, 70, 80, 90, 10, 20, 30*

(2 markah/*marks*)

c) Tulis sebuah program ujian untuk mengisi baris-gilir tersebut dengan nilai berikut [10, 20, 30, 40, 50, 60, 70, 80, 90]. Panggil metod ChangeOrder untuk mempamerkan implimentasi anda. Paparkan kesemua nilai tersebut.

Create a test program to fill the queue with the following values [10, 20, 30, 40, 50, 60, 70, 80, 90]. Call the *ChangeOrder* method to illustrate your implementation. Print all of the values.

(2 markah/*marks*)

* 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, dan antara aplikasi timbunan termasuklah untuk menyelesaikan masalah menterbalikkan sesuatu perkataan. Tuliskan suatu program menggunakan timbunan untuk menyelesaikan masalah tersebut. Sebagai contoh:

One popular technique in data structure is stack, and one of the many application of stact is to reverse a word. Write a program using stack to solve the problem. For example:

Andaikan input / *Say input*: MALAYSIA
Maka output / *Thus the output:* AISYALAM

Program anda mesti mempunyai metod-metod berikut:
Your program must contain the following methods:

i) Metod *push(char)* untuk menambah item di posisi teratas dalam timbunan.

Method push(char) to insert a new item to the top of the stack.

(2 markah/*marks*)

ii) Metod *pop()* untuk membuang item di posisi teratas dalam timbunan.

Method pop() to remove item from the top of the stack.

(2 markah/*marks*)

iii) Metod *peek()* yang memulangkan item teratas dalam timbunan tanpa membuangnya.

Method peek() that returns the top of the stack without removing it.

(2 markah/*marks*)

iv) Metod *isStackEmpty() yang memulangkan benar jika timbunan kosong.*

Method isStackEmpty() that returns true if the stack is empty.

(2 markah/*marks*)

v) Metod *isStackFull() yang memulangkan benar jika timbunan penuh.*

Method isStackEmpty() that returns true if the stack is full.

(2 markah/*marks*)

* 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. Tulis sebuah program untuk membina kelas senarai berpaut (tunggal) yang dipanggil *LinkedList*, di mana metod *main()* anda adalah seperti apa yang ditunjukkan di Rajah 1. *Output* anda adalah seperti apa yang ditunjukkan di Rajah 2.

   Write a program to create a (singly) linked list using a class called LinkedList, where your main() method is as shown in Figure 1. Your output is as shown on Figure 2.

```
public static void main(String a[]){
    SinglyLinkedListFinals sl = new SinglyLinkedListFinals();

    sl.add(10);
    sl.add(20);
    sl.add(30);
    sl.add(40);
    sl.add(50);
    sl.addAfter(11, 10);
    sl.addAfter(21, 20);
    sl.addAfter(31, 30);
    sl.addAfter(41, 40);
    sl.addAfter(51, 50);
    sl.traverse();
    sl.deleteFront();
    sl.deleteFront();
    sl.traverse();
    sl.deleteAfter(40);
    sl.deleteAfter(40);
    sl.deleteAfter(31);
    sl.deleteAfter(40);
    sl.traverse();

}
```

Rajah 1
Figure 1

```
run:
Adding: 10
Adding: 20
Adding: 30
Adding: 40
Adding: 50
Adding 11 after 10
Adding 21 after 20
Adding 31 after 30
Adding 41 after 40
Adding 51 after 50

Showing content of my linked list:
10 11 20 21 30 31 40 41 50 51

Deleting front: 10

Deleting front: 11

Showing content of my linked list:
20 21 30 31 40 41 50 51

Testing deleteAfter:
After 40 is 41. Deleting 41

Testing deleteAfter:
After 40 is 50. Deleting 50

Testing deleteAfter:
After 31 is 40. Deleting 40

Testing deleteAfter:
Element (40) not found...

Showing content of my linked list:
20 21 30 31 51
BUILD SUCCESSFUL (total time: 0 seconds)
```

Rajah 2
Figure 2

Program anda perlu mempunyai metod-metod berikut:

Your program requires the following methods:

i) Metod *add(int)* untuk memasukkan elemen baru ke dalam senarai berpaut.

Method add(int) to insert a new element into the linked list.

(3 markah/*marks*)

ii) Metod *addAfter(int, int)* untuk memasukkan elemen baru selepas elemen tertentu dalam senarai berpaut tersebut.

Method addAfter(int, int) to insert a new element after a particular element in the linked list.

(3 markah/*marks*)

iii) Metod *deleteFront()* untuk membuang elemen pertama di dalam senarai berpaut tersebut.
Method deleteFront() to delete the first element in the linked list.

(3 markah/*marks*)

iv) Metod *deleteAfter(int x)* untuk membuang sesuatu elemen selepas nilai *x* di dalam senarai berpaut.

Method deleteAfter(int x) to delete an element after element with value x in the linked list.

(3 markah/*marks*)

v) Metod *traverse()* untuk memaparkan kesemua kandungan dalam senarai berpaut tersebut.

Method traverse() to display all of the contents of the linked list.

(3 markah/*marks*)

* 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