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 II
ACADEMIC SESSION 2016/2017 : *SEMESTER II*

WIA1002 : Struktur Data
Data Structure

Jun 2017
June 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 yang dicetak)
(This question paper consists of 4 questions on 8 printed pages)

1. Tulis sebuah program untuk membina kelas senarai berpaut (tunggal) yang dipanggil *LinkedList*, di mana contoh output untuk program ujian anda adalah seperti berikut:

   Write a program to create a (singly) linked list using a class called LinkedList, where an example of the output for the test program is as follows:

```
run:

        Displaying the Linked List ********************************************
        Louis Vuitton: 10,000,000 Sold
        Chanel: 100,000,000 Sold
        Prada: 1,000,000 Sold
        Tods: 500,000,000 Sold

        Adding Coach after Chanel
        Found Chanel which is handbag number 2 in the linked list

        Displaying the Linked List ********************************************
        Louis Vuitton: 10,000,000 Sold
        Chanel: 100,000,000 Sold
        Coach: 10,000,000 Sold
        Prada: 1,000,000 Sold
        Tods: 500,000,000 Sold

        Adding Mulberry after Coach
        Found Coach which is handbag number 3 in the linked list

        Displaying the Linked List ********************************************
        Louis Vuitton: 10,000,000 Sold
        Chanel: 100,000,000 Sold
        Coach: 10,000,000 Sold
        Mulberry: 90,000,000 Sold
        Prada: 1,000,000 Sold
        Tods: 500,000,000 Sold

        Removing Chanel..
        Found a match.. Chanel is handbag number 2 in the linked list

        Displaying the Linked List ********************************************
        Louis Vuitton: 10,000,000 Sold
        Coach: 10,000,000 Sold
        Mulberry: 90,000,000 Sold
        Prada: 1,000,000 Sold
        Tods: 500,000,000 Sold

        Removing Coach..
        Found a match.. Coach is handbag number 2 in the linked list

        Displaying the Linked List ********************************************
        Louis Vuitton: 10,000,000 Sold
        Mulberry: 90,000,000 Sold
        Prada: 1,000,000 Sold
        Tods: 500,000,000 Sold

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

Program anda perlu mempunyai metod-metod berikut:

Your program requires the following methods:

i) Metod *insertFirstLink(String, int)* untuk memasukkan elemen baru ke dalam senarai berpaut.

Method insertFirstLink(String, int) to insert a new element into the linked list.

(2 markah/*marks*)

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

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

(4 markah/*marks*)

iii) Metod *removeLink(String)* untuk membuang mana-mana elemen di dalam senarai berpaut tersebut.

Method removeLink(String) to delete any element in the linked list.

(4 markah/*marks*)

iv) Metod *display()* untuk memaparkan kesemua kandungan dalam senarai berpaut tersebut.

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

(2 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.*

2. Bayangkan satu longgokan buku di atas meja anda. Setiap buku begitu besar dan berat dan anda hanya boleh keluarkan buku yang berada di atas longgokan itu. Anda tidak boleh mengeluarkan buku dari bawah satu sama lain. Begitu juga, anda boleh menambah buku lain dengan meletakkannya di atas bahagian atas longgokan. Anda tidak boleh menambah buku di bawah satu sama lain dan hanya boleh mengalih keluar buku bahagian atas longgokan itu.

   Imagine a pile of books on your desk. Each book is so large and heavy and you can only remove the top one from the pile. You cannot remove a book from under another one. Likewise, you can add another book to the pile only by placing it on the top of the pile. You cannot add a book beneath one another and can only remove the pile's top book.

   Mengambil kira semua maklumat yang diberikan di atas:

   Considering all of the given information above:

   a) Rekabentuk satu antaramuka tindanan, *StackInterface <E>* yang boleh anda gunakan untuk menentukan metod-metod biasa bagi apa-apa objek tindanan yang melaksanakan antaramuka ini. Antaramuka ini harus menentukan atribut dan metod biasa yang berikut:
      - Satu atribut yang merupakan satu *ArrayList* untuk menyimpan elemen-elemen dalam tindanan ini.
      - Satu konstruktor yang mencipta tindanan kosong.
      - Satu metod yang mengembalikan bilangan elemen dalam tindanan ini.
      - Satu metod yang mengembalikan elemen teratas dalam tindanan ini.
      - Satu metod yang mengembalikan dan membuang elemen teratas dalam tindanan ini.
      - Satu metod yang menambah elemen baru ke atas tindanan ini.
      - Satu metod yang mengembalikan benar jika tindanan adalah kosong.

      Design a stack interface, StackInterface<E> that you can use to specify common methods for any stack object that implements this interface. The interface should specify the following common attribute and methods:
      - *An attribute that is an ArrayList to store the elements in this stack.*
      - *A constructor that creates an empty stack.*
      - *A method that returns the number of elements in this stack.*
      - *A method that returns the top element in this stack.*
      - *A method that returns and removes the top element in this stack.*
      - *A method that adds a new element to the top of this stack.*
      - *A method that returns true if the stack is empty.*

      (2 markah/*marks*)

b) Jika anda mewakili buku-buku dengan tajuk mereka, rekabentuk satu kelas generik tindanan, *GenericStack <E>* yang melaksanakan *StackInterface<E>*, anda boleh menggunakannya untuk menjejak buku-buku dalam longgokan di atas meja anda. Laksanakan metod-metod dengan menggunakan *ArrayList*. Kelas ini perlu menggantikan metod Java *toString* untuk mengembalikan senarai elemen dalam tindanan ini.

If you represent books by their titles, design a stack generic class, GenericStack<E> that implement the StackInterface<E>, you can use it to track the books in the pile on your desk. Implement the methods using ArrayList. This class should override the Java toString method to return the list of elements in this stack.

(7 markah/*marks*)

c) Tulis satu program ujian, *TestPileStack* untuk menunjukkan pelaksanaan anda. Contoh output untuk program ujian ini adalah seperti berikut:

Write a test program, TestPileStack that demonstrates your implementation. An example of the output for the test program is as follows:

```
Create a new stack: an empty pile of books
isEmpty() returns true

Push 3 books to the pile:
Enter book title 1: The Hobbit
Enter book title 2: The Lord of the Rings
Enter book title 3: The Da Vinci Code

The new books that you added are: [The Hobbit, The Lord of the Rings, The Da Vinci
Code]

The pile should not be empty:
isEmpty() returns false
The pile has 3 books.

Get the top book and remove the top book:

The Da Vinci Code is at the top of the pile.
The Da Vinci Code is removed from the pile.

The Lord of the Rings is at the top of the pile.
The Lord of the Rings is removed from the pile.

The Hobbit is at the top of the pile.
The Hobbit is removed from the pile.

The pile should be empty:
isEmpty() returns true
```

(2 markah/*marks*)

3. Tulis kod bagi metod-metod berikut untuk digunakan di dalam sebuah kelas pepohon carian binari jenis E.

   Write a code for the following methods to be implemented in a generic binary search tree class type E.

   i) Metod *public boolean add(Node<E> node, E item)* untuk menambah item baru jenis E ke dalam pepohon carian binari. Metod tersebut perlu memulangkan nilai boolean: benar jika selitan berjaya, dan salah jika ianya item berulang.

   Method public boolean add(Node<E> node, E item) to add new item type E into the binary search tree. The method should return a boolean value: true if the insertion is successful, and false if it is a duplicated item.

   (3 markah/*marks*)

   ii) Metod *public void printTreeInOrder( )* untuk mencetak semua item-item pepohon dalam *in-order* secara iteratif (bukan-rekursif).

   Metod public void printTreeInOrder() to print all of the tree items in "in order" iteratively (non-recursive).

   (4 markah/*marks*)

   iii) Metod *public boolean contains(E item)* untuk menyemak jika pepohon carian binari mengandungi sesuatu item yang tertentu. Metod tersebut memulangkan nilai boolean: benar jika item dijumpai, dan salah sebaliknya.

   Method public boolean contains(E item) to check if the binary search tree contains a specified item. The method should return boolean value: true if the item is found, otherwise return false.

   (3 markah/*marks*)

4. Tulis satu aturcara yang menerima nombor dan mengekodkannya untuk mengembalikan perkataan sepadan dengan menggunakan baris-gilir.

   Write a program to accept numbers and encode them to return their corresponding word by using queue.

   Program anda harus menunjukkan perkara berikut:

   Your program shall demonstrate the following:

   a) Laksanakan sebuah kelas *Queue<E>*. Pastikan metod-metod dan konstruktor-konstruktor berikut dilaksanakan.

   Implement a Queue<E> class. Ensure that the following methods and constructors are implemented.

i) `public Queue(E[] items)`

ii) `public Queue()`

iii) `public void enqueue(E e)`

iv) `public E dequeue()`

v) `public E getElement(int i)`

vi) `public int getSize()`

vii) `public boolean isEmpty()`

viii) `public String toString()`

(8 markah/*marks*)

b) Aturcara tersebut menggunakan kelas *Queue<E>* yang dilaksanakan di dalam Soalan 4(a). Berikut adalah maklumat lanjut bagi membantu pembangunan aturcara tersebut:

The program uses the Queue<E> class implemented in Question 4(a). Following are further hints to guide the program development:

i) Senarai huruf (i.e., a - z) diberikan sebagai satu tatasusun seperti berikut. Simpan tatasusun huruf ini dalam baris-gilir yang telah dilaksanakan dalam Soalan 4(a). Terdapat sejumlah 26 aksara.

A list of alphabets (i.e., a - z) is given as an array as follows. Store this array of alphabets in the queue implemented in Question 4(a). There are 26 characters in total.

```
alphabet = {'a','b','c',.........,'y','z'}
```

ii) Indeks (lihat contoh output) mewakili lokasi indeks bagi huruf dan tidak diberikan sebagai tatasusun berasingan.

The index (see example output) represents the index location of the alphabets and not given as a separate array.

iii) Program ini berinteraksi dengan pengguna melalui pertanyaan mengenai frekuensi input nombor yang akan diberikan.

The program interacts with the user by asking the frequency of input numbers to be provided.

iv) Setelah pengguna memasukkan semua nombor, program ini akan memaparkan huruf yang sepadan.

Once the user enters all the numbers, the program will display their corresponding alphabets.

(9 markah/*marks*)

Berikut adalah contoh output. Teks yang berhuruf tebal adalah input daripada pengguna.

Following are examples of the output. The bolded texts are the user input.

Contoh Output 1:
Example Output 1:

```
Queue: [a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z]
Index: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
26]

How many times will you enter a number: 0
(Please enter your number(s) between 0-26.)
The entered numbers are []
The deciphered values are -Cannot decipher. No value was entered.-
```

Contoh Output 2:
Example Output 2:

```
Queue: [a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z]
Index: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
26]

How many times will you enter a number: 13
(Please enter your number(s) between 0-26.)
Enter number 1 >> 3
Enter number 2 >> 0
Enter number 3 >> 19
Enter number 4 >> 0
Enter number 5 >> 18
Enter number 6 >> 19
Enter number 7 >> 17
Enter number 8 >> 20
Enter number 9 >> 2
Enter number 10 >> 19
Enter number 11 >> 20
Enter number 12 >> 17
Enter number 13 >> 4
The entered numbers are [3, 0, 19, 0, 18, 19, 17, 20, 2, 19, 20, 17, 4]
The deciphered values are datastructure
```

TAMAT
END