UNIVERSITI MALAYA
UNIVERSITY OF MALAYA

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

SESI AKADEMIK 2015/2016 : SEMESTER II
ACADEMIC SESSION 2015/2016 : *SEMESTER II*

WIA1002 : Struktur Data
Data Structure

Jun 2016
June 2016

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 9 halaman bercetak)
(This question paper consists of 4 questions on 9 printed pages)

1. Satu minggu mengandungi tujuh hari. Sebuah kelas yang dipanggil Hari harus menyimpan hari dari Isnin sehingga Ahad. Reka dan tulis sebuah program untuk hari dalam minggu yang perlu melakukan perkara berikut:

   A week consists of seven days. A class called Day should store the days from Monday through Sunday. Design and write a program for days of the week that should do the following:

   i) Membolehkan pengguna untuk memasukkan satu input bagi hari berdasarkan nombor seperti 1 untuk Isnin dan 2 untuk Selasa. (i.e., 1 = Isnin, 2 = Selasa, ... , 7 = Ahad).

      Allows user to key-in an input of the day based on numbers such as 1 for Monday and 2 for Tuesday. (i.e., 1 = Monday, 2 = Tuesday, ... , 7 = Sunday).

   ii) Paparkan nama penuh yang sepadan bagi hari (i).

      Display the corresponding full name of the day of (i).

   iii) Kira dan paparkan satu hari berikut dan satu hari sebelumnya berdasarkan nombor yang dimasukkan dalam (i).

      Calculate and display the following and previous days based on the number entered in (i).

   iv) Kira dan kembalikan hari dengan menambah hari-hari tertentu untuk hari semasa sebagaimana yang dinyatakan oleh (ii). Sebagai contoh, jika hari semasa adalah Isnin dan kita menambah empat hari, hari yang dikembalikan adalah hari Jumaat. Begitu juga, jika hari semasa adalah Selasa dan kita menambah 13 hari, hari yang dikembalikan adalah Isnin.

      Calculate and return the day by adding certain days to the current day as specified by (ii). For example, if the current day is Monday and we add four days, the day to be returned is Friday. Similarly, if the current day is Tuesday and we add 13 days, the day to be returned is Monday.

   Dengan mengambil kira semua maklumat yang diberikan di atas:

   Considering all of the above given information:

   a) Kenalpasti keperluan program tersebut dengan mewakili mereka menggunakan satu Kelas UML untuk kelas Hari.

      Identify the program requirements by representing them using one UML Class for the Day class.

      (3 markah/*marks*)

   b) Tulis sebuah program untuk melaksanakan kelas Hari berdasarkan keperluan yang telah dikenalpasti.

      Write a program to implement the Day class following the identified requirements.

      (5 markah/*marks*)

c) Tulis sebuah program ujian mengikut format yang diberikan dalam contoh output berikut.

Write a test program in the format given in the following output examples.

(2 markah/*marks*)

Output contoh program ini adalah seperti berikut. Teks dalam **huruf tebal** adalah input pengguna.

*An example output of the program is as follows. The text in **bold** is the user input.*

```
Specify your day in number:
1 for Monday
2 for Tuesday
3 for Wednesday
4 for Thursday
5 for Friday
6 for Saturday
7 for Sunday
1
The name of the day is : Monday
The following day of Monday is : Tuesday
The previous day of Monday is : Sunday
How many days to add to the specified/current day (Monday)? :
4
The new day after the addition of 4 day(s) is : Friday
```

Rajah 1: Contoh Output 1
Figure 1: Output Example 1

```
Specify your day in number:
1 for Monday
2 for Tuesday
3 for Wednesday
4 for Thursday
5 for Friday
6 for Saturday
7 for Sunday
7
The name of the day is : Sunday
The following day of Sunday is : Monday
The previous day of Sunday is : Saturday
How many days to add to the specified/current day (Sunday)? :
7
The new day after the addition of 7 day(s) is : Sunday
```

Rajah 2: Contoh Output 2
Figure 2: Output Example

```
Specify your day in number:
1 for Monday
2 for Tuesday
3 for Wednesday
4 for Thursday
5 for Friday
6 for Saturday
7 for Sunday
4
The name of the day is : Thursday
The following day of Thursday is : Friday
The previous day of Thursday is : Wednesday
How many days to add to the specified/current day (Thursday)?
:
23
The new day after the addition of 23 day(s) is : Saturday
```

Rajah 3: Contoh Output 3
Figure 3: Output Example 3

2. Dari perspektif seorang pengaturcara, salah satu tugas yang paling mencabar adalah untuk memadankan kurungan kiri dengan kurungan kanan. Kurungan-kurungan yang lazimnya digunakan dalam menulis program tersenarai dalam Jadual 1.

   From the perspective of a programmer, one of the most challenging tasks is to match the left-bracket to the corresponding right-bracket. Some of the commonly used brackets in writing a program are listed in Table 1.

Jadual 1: Kurungan-kurungan yang lazimnya digunakan
Table 1: Brackets that are commonly in use

Kiri / *Left*	Kanan / *Right*	Nama / *Name*
[	]	*Bracket*
{	}	*curly bracket / curly braces*
(	)	*round bracket / parenthesis*
<	>	*angled bracket / chevron*

   a) Rangka satu algoritma untuk memeriksa sama ada input jenis *string* yang dibekalkan terdiri daripada kurungan-kurungan yang telah dipadankan. Algoritma itu akan memberi output "YA – semua kurungan telah dipadankan!" apabila semua kurungan kiri dipadankan oleh kurungan kanan mengikut jenis yang berpadanan. Sekiranya ada kurungan yang tidak disepadankan, algoritma itu akan memberikan output "*Not all brackets are matched*". Jika tidak ada kurungan dalam input jenis *string*, output "*There is no bracket in the input*". Ringkaskan algoritma anda dengan menggunakan format kod-pseudo.

Devise an algorithm to check whether an input of type string consists of matched brackets. The algorithm outputs "YES - all matched!" when all left-brackets are matched by their corresponding right-brackets of the same type. If there exists some brackets that not matched, the algorithm should output "Not all brackets are matched". If there is no bracket in the input string, output "There is no bracket in the input". Summarize your algorithm in the form of pseudo-code.

(5 markah/*marks*)

b) Terjemahkan kod-pseudo anda ke dalam sebuah program yang menerima input berjenis *string* melalui teks fail yang bernama – *BracMatch.txt*. Fail tersebut mengandungi beberapa barisan teks. Contoh maklumat yang terkandung di dalam fail *BracMatch.txt* adalah seperti berikut:

Translate your psuedo-code into a program, which accepts the input string through a text file called - BracMatch.txt. The file consists of several lines of texts. An example of the content of the BracMatch.txt file is as follows:

(x,y;z)
(area<un[der{the}curv]e>nothing)
(DS-{is[my>]}
][
datastructure

(5 markah/*marks*)

3. 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:

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 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*

(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, jika *k=4*, maka output seharusnya 40,50,60,70,80,90,10,20,30.

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, if k=4, then the output should be 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*)

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 4. ***Output*** anda adalah seperti apa yang ditunjukkan di Rajah 5. Program anda perlu mempunyai metod-metod berikut:

*Write a program to create a (singly) linked list using a class called LinkedList, where your **main()** method is as shown in Figure 4. Your **output** is as shown on Figure 5. Your program requires the following methods:*

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

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.*

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

    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 4
Figure 4

```
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 5
Figure 5

TAMAT
END