UNIVERSITI MALAYA
UNIVERSITY OF MALAYA

UJIAN ATAS TALIAN IJAZAH SARJANA MUDA SAINS KOMPUTER / SARJANA MUDA TEKNOLOGI MAKLUMAT
ONLINE TEST FOR THE DEGREE OF BACHELOR OF COMPUTER SCIENCE / BACHELOR OF INFORMATION TECHNOLOGY

SESI AKADEMIK 2021/2022 : SEMESTER II
ACADEMIC SESSION 2021/2022 : SEMESTER II

WIA1002/WIB1002 : Struktur Data
Data Structure

Jun 2022 Masa: 2 jam
June 2022 Time: 2 hours

ARAHAN KEPADA CALON :
INSTRUCTIONS TO CANDIDATES :

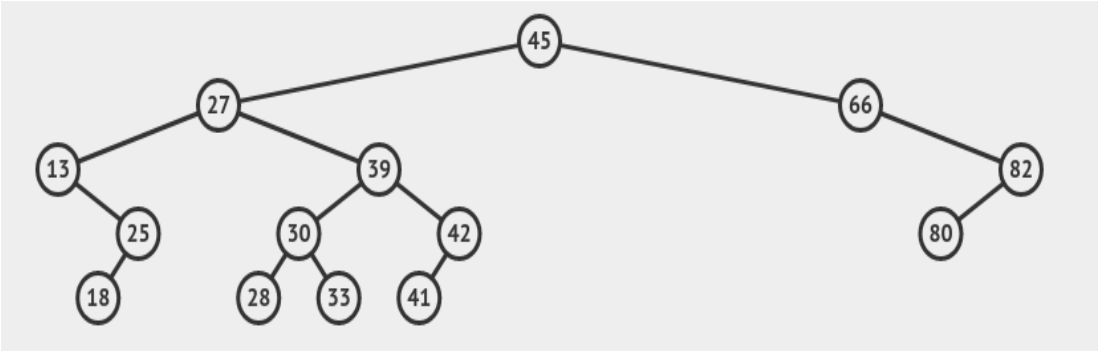
Jawab SEMUA soalan (50 markah).

Answer ALL questions (50 marks).

(Kertas soalan ini mengandungi 6 soalan dalam 13 halaman yang dicetak)
(This question paper consists of 6 questions on 13 printed pages)

1) Rajah 1.1 menunjukkan satu pokok carian binari.

Figure 1.1 shows a binary search tree.



Rajah 1.1: Pokok carian binary
Figure 1.1: Binary search tree

a) Tulis urutan node-node jika pokok dilintasi dengan:

Write the sequence of the nodes if the tree is traversed with:

i. lintasan *inorder* / *inorder traversal*
ii. lintasan *postorder* / *postorder traversal*

(4 markah/marks)

b) Lukis semula pokok jika nod 66 dan 27 dipadam.

Redraw the tree if nodes 66 and 27 are deleted.

(2 markah/*marks*)

2) Tulis program yang mengandungi kaedah generik yang membalikkan susunan unsur-unsur dalam struktur data *stack* menggunakan rekursi. Di samping itu, tulis metod *main* untuk menguji kaedah static tersebut. Dua set data ujian harus digunakan dalam metod *main*: i) {"Switch", "Motherboard", "RAM", "SSD", "CPU", "GPU", "Router"}; dan ii) {17, 21, 45, 23, 1, 99, 16}. Tiada markah akan diberikan jika konsep *stack* dan rekursi tidak digunakan dalam program ini.

Write a program that contains a generic method that reverses the order of the elements within a stack data structure using recursion. In addition, write the main method to test the static method. Two sets of test data should be used in the main method: i) {"Switch", "Motherboard", "RAM", "SSD", "CPU", "GPU", "Router"}; and ii) {17, 21, 45, 23, 1, 99, 16}. No marks will be awarded if the stack and recursion concepts are not applied in the program.

(5 markah/*marks*)

3) Untuk mengelakkan penularan *coronavirus*, Klinik Maya hanya boleh mengendalikan 10 lawatan pesakit setiap hari. Klinik ini ingin menggunakan baris gilir generik yang dilaksana dengan menggunakan *Java ArrayList* untuk tujuan ini. Menurut Ketua Pegawai Teknologi klinik ini, salah satu faedah menggunakan *ArrayList* ialah mereka boleh menyimpan semua maklumat pesakit sehingga penghujung hari. Rajah 3.1 menunjukkan pelaksanaan yang tidak lengkap bagi kelas baris gilir:

To prevent the spread of coronavirus, Maya Clinic only can handle 10 patient visits per day. The clinic wants to use a generic queue implemented using the Java ArrayList. According to the Chief Technology Officer of the clinic, one benefit of using ArrayList is that they can keep all the patient information until the end of the day. Figure 3.1 shows an incomplete implementation of the queue class:

```
import java.util.ArrayList;
public class ArrayListQueue<E> {
    private int size=10;
    private int front=0;
    private int end=_________;  // subquestion (a)
    private ArrayList<E> list = new ArrayList<E>(size);

    public boolean enqueue (E o) {
        boolean success=false;
        if (end < size)    {
            success=list.add(o);
            ___________              // subquestion (b)
        }
        return success;
    }

    public E dequeue () {        // subquestion (c)

    }

    public String queueToString() {  // subquestion (d)

    }

    public String allToString() {
        return "ALL : " + list.toString();
    }
}
```

Rajah 3.1: Pelaksanaan yang tidak lengkap bagi kelas baris gilir
Figure 3.: An incomplete implementation of the queue class

a) Lengkapkan definasi pemboleh ubah "*end*" yang merupakan indeks kepada elemen terakhir di dalam baris gilir.

Complete the definition of variable "end", which is the index of the last element in the queue.

(1 markah/mark)

b) Tulis satu baris kod supaya metod `enqueue` lengkap. Kaedah ini seharusnya memasukkan satu objek ke kedudukan yang sesuai di dalam *ArrayList*.

Write one line of code so that the method `enqueue` *complete. The method should insert an object to an appropriate position in the ArrayList.*

(1 markah/mark)

c) Lengkapkan metod `dequeue` yang mengembalikan objek pertama yang belum dirujuk dalam baris gilir. Ambil perhatian bahawa selepas pelaksanaan metod ini, objek yang telah dikembalikan masih harus disimpan dalam senarai *ArrayList* dan boleh diakses oleh metod `allToString`. Jika tiada objek untuk dikembalikan, metod ini mengembalikan *null*.

Complete the `dequeue` *method, which returns the first object which has not been referred to in the queue. Take note that after this method's execution, the returned object should still be stored in the ArrayList list and can be accessed by the* `allToString` *method. If there is no object to return, this method returns a null.*

(2 markah/marks)

d) Lengkapkan metod `queueToString` yang mengembalikan semua objek dalam baris gilir dalam urutan sebagai rentetan. Objek-objek yang telah dialih keluar dari baris gilir tidak boleh dikembalikan.

Complete the `queueToString` *method, which returns all the objects in the queue in sequence as a string. The objects which have been removed from the queue should not be returned.*

(2 markah/*marks*)

4) Sebuah syarikat akan mengadakan cabutan bertuah semasa makan malam tahunan mereka. Pada masa ini, syarikat ini hanya ada 10 orang pekerja dan semuanya layak menyertai aktiviti ini jika mereka menghadiri makan malam tersebut. Tetapi, senarai akhir penyertaan makan malam masih belum muktamad lagi.

Untuk memastikan cabutan bertuah ini berjaya, syarikat ini hendak menulis sebuah aturcara supaya cabutan bertuah ini memilih pekerja bertuah secara rawak dari senarai yang diberi. Diberi bahawa pengaturcara yang bertanggungjawab sudah mempunyai tiga fail berikut:

A company will have a lucky draw activity during its annual dinner. Currently, the company only has 10 staff and all are eligible to join the activity if they attend the dinner. However, the final list of joining the dinner is not confirmed yet.

To ensure that the lucky draw is successful, the company wants to write a program so that lucky staff are selected randomly from a list provided. Given that the programmer in charge of the task already has the following three files:

Staff.java

```
public class Staff {
    private int staffID;
    private String staffName;
    private String position;

    public Staff() {
    }

    public Staff(int id, String name, String post) {
      staffID= id;
      staffName = name;
      position = post;
    }

    public int getID () {
      return staffID;
    }

    public String getName () {
      return staffName;
    }

    public String getPosition () {
      return position;
    }
}
```

Node.java

```
public class Node<E> {
    E element;
    Node<E> next;

    public Node(E element) {
      this.element = element;
    }
}
```

LinkedList.java

```
public class LinkedList<E> {
  private Node<E> head, tail;
  private int size=0;

  public LinkedList() {   }

  public int getSize() {
```

```
    return size;
  }

  public void addLast(E e) {
    if (tail == null)  head = tail = new Node<>(e);
    else {
      tail.next = new Node<>(e);
      tail = tail.next;
    }
    size++;
  }

  public E removeFirst() {
    if (size == 0) return null;
    else {
      Node<E> temp = head;
      head = head.next;
      size--;
      if (head == null) tail = null;
      return temp.element;
    }
  }

  public E removeLast() {
    if (size == 0) return null;
    else if (size == 1) {
      Node<E> temp = head;
      head = tail = null;
      size = 0;
      return temp.element;
    }
    else {
      Node<E> current = head;
      for (int i = 0; i < size - 2; i++)
        current = current.next;
      Node<E> temp = tail;
      tail = current;
      tail.next = null;
      size--;
      return temp.element;
    }
  }

  public E remove(int index) {
    if (index < 0 || index >= size) return null;
    else if (index == 0) return removeFirst();
    else if (index == size - 1) return removeLast();
    else {
      Node<E> previous = head;
      for (int i = 1; i < index; i++)
```

```
        previous = previous.next;
      Node<E> current = previous.next;
      previous.next = current.next;
      size--;
      return current.element;
    }
  }
}
```

Untuk memastikan aturcara cabutan bertuah ini rawak dan adil, kamu telah dilantik sebagai pengaturcara luaran untuk melengkapkan tugas-tugas berikut:

To make sure that the lucky draw program is random and fair, you are appointed as an external programmer to complete the following tasks:

a) Membangunkan fail keempat "*LuckyDraw.java*" dan tambah semua pekerja di bawah kepada satu senarai pautan bernama "*staffList1*", seperti yang disenarai dalam Jadual 4.1.
(catatan: satu fail yang mengandungi senarai ini (*staffList.txt*) telah diberi)

Create the fourth file, "LuckyDraw.java", and add all the following staff to a linked-list called "staffList1", as shown in Table 4.1.

(note: a file with the list (staffList.txt) is given)

Staff ID	*Staff Name*	*Position*
1	*Luke Skywalker*	*Manager*
2	*Han Solo*	*Supervisor*
6	*Moff Tarkin*	*Assistant*
8	*Obi Wan*	*Assistant*
9	*Organa*	*Assistant*
12	*Leia Organa*	*Supervisor*
15	*Chewbacca*	*Assistant*
16	*Uncle Owen*	*Assistant*
17	*Aunt Beru*	*Assistant*
19	*Lando Calrissian*	*Assistant*

Jadual 4.1: Senarai pekerja
Table 4.1: Staff list

(1 markah / *mark*)

b) Dalam fail "*LinkedList.java*", tambah satu metod "*get*" yang menerima satu parameter nombor bulat sebagai indeks objek dalam senarai pautan, dan memulangkan objek berjenis generik pada indeks tersebut. Metod ini tidak seharusnya memanggil metod lain dalam senarai pautan.

In the file "LinkedList.java", add one method "get" that receives an integer parameter as the index of an object in the linked-list, and returns the object of generic type at the given index. This method should not call any other methods in the linked-list.

(2 markah/*marks*)

c) Dalam fail "*LinkedList.java*", tambah satu metod "*clone*" yang tidak menerima sebarang parameter. Metod ini mencipta dan memulangkan satu salinan senarai pautan yang memanggilnya.

In the file "LinkedList.java", add one method "clone" that receives no parameter. This method creates and returns a copy of the linked-list object that calls it.

(*2 markah/marks*)

Pihak pengurusan telah memutuskan untuk menjalankan 2 pusingan cabutan bertuah. Hadiah untuk pusingan pertama ialah 5 dron dari model yang sama, manakala pusingan kedua ialah baucar tunai yang jumlah nilainya RM1000.

The management has decided to have 2 rounds of lucky draw. The prizes in the first round are 5 drones of the same model, whereas in the second round are cash vouchers with a total value of RM1000.

d) Untuk penyediaan aktiviti ini, buat satu salinan "*staffList1*" dan namakannya "*staffList2*" di dalam fail "*LuckyDraw.java*".

To prepare for the activity, make a copy of "staffList1" and name it as "staffList2" in the file "LuckyDraw.java".

(1 markah / *mark*)

e) Untuk pusingan pertama, pilih 5 pekerja secara rawak dari "*staffList1*" untuk mendapatkan hadiah. Ambil perhatian bahawa setiap pekerja tidak boleh memenangi lebih daripada 1 hadiah dalam pusingan ini. Tulis kod untuk tugas ini dan keluaran dijangka seperti format di dalam Rajah 4.1. Kecekapan penyelesaian adalah kritikal.

For the first round, randomly pick 5 staff from the "staffList1" for the prizes. Take note that every staff cannot win more than 1 prize in this round. Write code for this task and the output in the following format is expected as in Figure 4.1. The efficiency of the solution is critical.

```
8 Obi Wan wins a drone!
9 Organa wins a drone!
6 Moff Tarkin wins a drone!
1 Luke Skywalker wins a drone!
12 Leia Organa wins a drone!
```

Rajah 4.1: Keluaran dijangka untuk soalan 4e)
Figure 4.1: expected output for question 4e)

(4 markah/*marks*)

f) Untuk pusingan kedua, guna "*staffList2*". Semua orang, termasuk yang telah memenangi hadiah dalam pusingan pertama layak menyertainya. Bukan setahap ini sahaja, tetapi setiap orang boleh memenangi lebih daripada satu

hadiah dalam pusingan ini. Baucar tunai dengan jumlah nilainya RM1000 akan diberi dengan mengikut peraturan-peraturan berikut:

- Jika pemenang seorang pembantu, pemenang akan dapat baucar RM200.
- Jika pemenang seorang penyelia, pemenang akan dapat baucar RM100.
- Jika pemenang seorang pengurus, namanya akan diumumkan tetapi tiada baucar tunai akan diberi.
- Cabutan bertuah akan berakhir jika baucar tunai habis diberi. Untuk pemenang terakhir, jika baucar tunai yang tertinggal kurang daripada jumlah yang dia sepatutnya terima, pemenang ini hanya dapat jumlah yang tertinggal sahaja.

Tulis kod untuk tugas ini dan keluaran dalam format di dalam Rajah 4.2 dijangka:

For the second round, use "staffList2" instead. Everyone, including those who have won a prize in the first round, can join. Not only that, everyone is allowed to win more than one prize in this round. The cash vouchers with a total value of RM1000 will be given out following the rules:

- *If the winner is an assistant, the winner will get an RM200 voucher*
- *If the winner is a supervisor, the winner will get an RM100 voucher*
- *If the winner is a manager, the name will be announced, but no cash voucher will be given.*
- *The lucky draw ends when all the cash vouchers have been given out. For the last winner, if the cash voucher left is less than the amount he/she is entitled to, this winner will get the amount left only.*

Write code for this task, and the output in the format as shown in Figure 4.2 is expected:

```
Assistant Uncle Owen wins RM200. Cash vouchers left RM 800
Assistant Aunt Beru wins RM200. Cash vouchers left RM 600
Manager Luke Skywalker is the winner, but no cash voucher will be given
Supervisor Leia Organa wins RM100. Cash vouchers left RM500
Supervisor Han Solo wins RM100. Cash vouchers left RM400
Supervisor Han Solo wins RM100. Cash vouchers left RM300
Assistant Uncle Owen wins RM200. Cash vouchers left RM 100
Assistant Obi Wan wins RM100. Cash vouchers left RM 0
```

Rajah 4.2: Keluaran dijangka untuk soalan 4f)
Figure 4.2: Expected output for question 4f)

(4 markah/*marks*)

5) Tukarkan ungkapan *infix* berikut kepada *postfix*:

Convert the following infix expressions to postfix:

a) $a + b - c \times \frac{d}{e}$

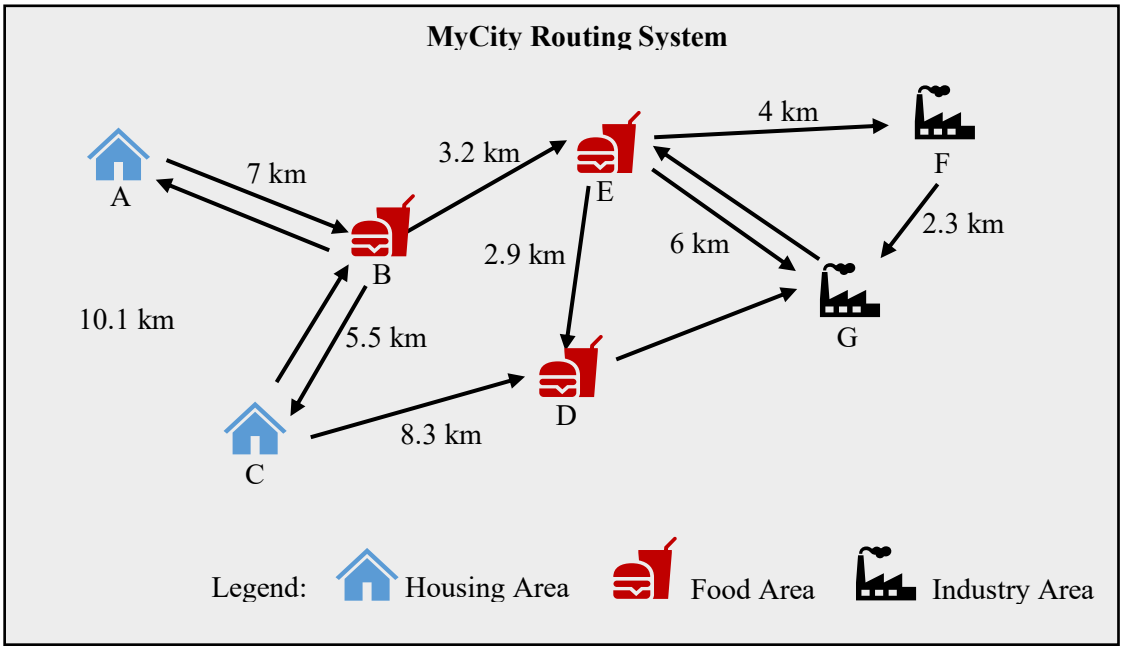
b) $\frac{a}{b} - c \times (d + e)$

c) $a \times b + \frac{c \times d}{e} - f + (g - h)$

(3 markah/*marks*)

6) Anda baru mengesahkan tawaran pekerjaan daripada Industri G dalam bandar MyCity dan anda telah memindah ke A Kawasan Perumahan untuk dekat kepada tempat kerja anda. Tetapi, bandar MyCity adalah kawasan baru kepada anda, jadi anda telah mendapatkan peta daripada majlis perbandaran seperti ditunjukkan dalam Rajah 6.1 untuk lebih mengetahui tempat ini.

You have just confirmed a new job offer from G Industry in MyCity town and you have moved to Housing Area A to near to your new workplace. However, MyCity town is a new area to you and hence you obtained the map from the town council as shown in Figure 6.1 to get to know this place better.



Rajah 6.1: Sistem penghalaan MyCity.
Figure 6.1: MyCity routing system.

Dalam pejabat majlis perbandaran, pengawai tersebut dengan baik hati mengkongsikan dengan anda anggaran kelajuan memandu yang anda akan gunakan semasa memandu di waktu pagi yang sibuk dalam pelbagai kawasan:

- Kawasan Perumahan ⇆ Kawasan Perumahan = y min/km
- Kawasan Perumahan ⇆ Kawasan Pemakanan = y min/km
- Kawasan Pemakanan ⇆ Kawasan Pemakanan = 1.25y min/km
- Kawasan Pemakanan ⇆ Kawasan Industri = 1.5y min/kim
- Kawasan Industri ⇆ Kawasan Industri = 3y min/km

Sebagai contoh, kalau anda memandu daripada A ke B, andaikan $y = 2$, **jarak memandu** adalah 7 km dan **tempoh masa memandu** adalah 14 minit kerana **kelajuan memandu** dalam Kawasan Perumahan adalah berkelajuan y.

In the town council office, the officer also kindly shared with you the estimated driving speed that you will be used to commute in the busy morning within various areas:

- *Housing Area ⇆ Housing Area = y min/km*
- *Housing Area ⇆ Food Area = y min/km*

- *Food Area* ⇆ *Food Area* = $1.25y$ *min/km*
- *Food Area* ⇆ *Industry Area* = $1.5y$ *min/km*
- *Industry Area* ⇆ *Industry Area* = $3y$ *min/km*

For example, if you are commuting from A to B, assuming that $y = 2$*, the* ***commute distance*** *will be 7 km and the* ***commute duration*** *will be 14 minutes because the* ***driving speed*** *within these housing areas is* y *speed.*

Untuk lebih memahami keadaan trafik semasa waktu pagi, anda memutuskan bahawa anda akan membangunkan sebuah program seperti yang berikut:

To better understand the traffic situation during the morning commute, you have decided to develop a program as follows:

a) Bina satu graf untuk menangkap system penghalaan dalam bandar MyCity. Khususnya, Titik A hingga Titik G akan dijadikan *Vertexes* manakala penghalaan dalam titik-titik adakan dijadikan *Edges*. Dalam kelas **Edge** anda, anda perlukan menangkap kelajuan memandu dan jarak memandu, secara berasingan.

   Anda juga dikehendaki untuk menyediakan metod-metod utiliti yang berikut dalam kelas **RoutingGraph** anda:

   i) `getSize` untuk menunjukkan bilangan *Vertexes*.
   ii) `getVertex` untuk mendapatkan nama Titik.
   iii) `hasEdge` untuk memastikan kewujudan *Edge* antara dua *Vertexes*.
   iv) `getNeighbours` untuk menyenaraikan semua penyambung *Vertex(es).*
   v) `printEdges` yang akan mencetak semua *Edges* dengan kelajuan memandu dan jarak memandu masing-masing.

   Lepas itu, bina satu kelas **TestGraph** dan tuliskan metod *main* untuk membina graf dan memaparkan maklumat yang berikut seperti dalam Rajah 6.2 dengan menggunakan metod-metod utiliti (mengandaikan $y = 2$):

   Create a graph to capture the routing system of MyCity town. In particular, points A to G will be the Vertexes, while the routes within these points will be the Edges. In your **Edge** *class, you should separately capture the driving speed and commute distance.*

   You are also required to prepare the following utility methods in your **RoutingGraph** *class:*

   i) `getSize` *to display the number of Vertexes.*
   ii) `getVertex` *to obtain the Point name.*
   iii) `hasEdge` *to identify whether there is an Edge between two Vertexes.*
   iv) `getNeighbours` *to list all connecting Vertex(es).*
   v) `printEdges` *that will print all the Edges with their respective driving speed and commute distance.*

Then, create a **TestGraph** *class and write the main method to create the graph and display the following information as shown in Figure 6.2 using the utility methods (assume that* $y = 2$*):*

```
The number of vertices in MyCityGraph: 7
List all vertices:
0: A    1: B    2: C    3: D    4: E    5: F    6: G
Has edge from B to A?  true
Has edge from A to D?  false

Find all neighbours of B : [E, C, A]

Print all edges :
# A : [A,C(speed=1.0 , distance=10.1)] [A,B(speed=1.0 , distance=7.0)]
# B : [B,E(speed=1.25 , distance=3.2)] [B,C(speed=1.0 , distance=5.5)] [B,A(speed=1.0 , distance=7.0)]
# C : [C,D(speed=1.25 , distance=8.3)] [C,B(speed=1.0 , distance=5.5)]
# D : [D,G(speed=1.5 , distance=4.9)]
# E : [E,F(speed=1.5 , distance=4.0)] [E,G(speed=1.5 , distance=6.0)] [E,D(speed=1.25 , distance=2.9)]
# F : [F,G(speed=1.2 , distance=2.3)]
# G : [G,E(speed=1.5 , distance=6.0)]
```

Rajah 6.2: Keluaran untuk soalan 6a).
Figure 6.2: Output for question 6a).

(4 markah/*marks*)

b) Senaraikan **LIMA (5)** laluan yang boleh digunakan untuk memandu daripada Titik A ke Titik G. Setiap *Vertex* hanya boleh digunakan satu kali untuk mengelakkan gelung.

List ***FIVE (5)*** *possible paths to commute from Point A to Point G. Each Vertex should only be utilized once to avoid a loop.*

(2 markah/*marks*)

c) Dalam kelas **TestGraph** anda, bina dua metod bernama metod `calculateDistance` dan metod `calculateDuration` untuk mengira jumlah jarak dan tempoh masa untuk semua laluan yang disenaraikan dalam Soalan (b). Lepas itu, dalam metod *main*, panggil metod-metod tersebut dan tunjukkan keluaran berikut seperti dalam Rajah 6.3:

In your ***TestGraph*** *class, create two methods namely* `calculateDistance` *method and* `calculateDuration` *method to compute the distances and commute duration for all the paths identified in Question (b). Then, in the main method, call the methods and show the output as in Figure 6.3:*

```
Path 1: Your Path 1
Distance= 25.70 km, Duration= 60.45 min
Path 2: Your Path 2
Distance= 16.20 km, Duration= 40.00 min
Path 3: Your Path 3
Distance= 25.10 km, Duration= 56.72 min
Path 4: Your Path 4
Distance= 18.00 km, Duration= 43.95 min
Path 5: Your Path 5
Distance= 23.30 km, Duration= 55.65 min
```

Rajah 6.3: Keluaran untuk soalan 6c). Jarak dan tempoh masa anda mungkin berbeza bergantung kepada laluan yang telah disenaraikan.
Figure 6.3: Output for question 6c). Depending on your listed paths, your distances and durations might differ from the shown figure.

(6 markah/*marks*)

d) Dalam kelas **TestGraph** anda, bina dua lagi metod untuk menyusun laluan mengikut jarak dan tempoh masa yang didapatkan. Anda adalah dikehendaki untuk membina metod `sortDistance` untuk menyusun laluan dengan jarak yang didapati dalam susunan menaik melalui bubble sort dan metod `sortDuration` untuk menyusun laluan dengan tempoh masa yang didapati dalam susunan menurun melalui insertion sort. Lepas itu, dalam metod main, panggil dua metod ini dan tunjukkan keluaran yang berikut:

*In your **TestGraph** class, create another two methods to sort the paths based on their computed distance and commute duration. You must create a* `sortDistance` *method to sort the paths using the computed distance in ascending order via bubble sort and a* `sortDuration` *method to sort the paths using the computed commute duration in descending order via insertion sort. Then, in the main method, call these two methods and show the following output:*

```
After Bubble Sort:
Path 2 (16.20 km)
Path 4 (18.00 km)
Path 5 (23.30 km)
Path 3 (25.10 km)
Path 1 (25.70 km)


After Insertion Sort:
Path 1 (60.45 min)
Path 3 (56.72 min)
Path 5 (55.65 min)
Path 4 (43.95 min)
Path 2 (40.00 min)
```

Rajah 6.4: Keluaran untuk Soalan 6d). Keputusan menyusun, jarak dan tempoh masa anda mungkin berbeza bergantung kepada laluan yang telah disenaraikan.
Figure 6.4: Output for question 6d). Depending on your listed paths, your sorting results, distances, and durations might differ from the shown figure.

(4 markah/*marks*)

TAMAT
END