Jawab semua soalan menggunakan **Netbeans** dan hantar semua jawapan anda berdasarkan arahan yang diberikan oleh pengawas peperiksaan.

*Answer all questions using **Netbeans** and submit all answers according to the instructions given by the invigilators.*

1. **Senarai Terpaut** adalah struktur data linear di mana elemen (dipanggil nod) disimpan secara tidak berturutan dalam memori. Setiap nod mengandungi data dan penunjuk. Berbeza dengan *array*, senarai terpaut tidak memerlukan saiz yang telah ditetapkan dan boleh berkembang atau mengecil secara dinamik. Jenis-jenis senarai terpaut termasuk **senarai** terpaut **tunggal** dan **senarai** terpaut **ganda**.

   *A **Linked List** is a linear data structure where elements (called nodes) are stored non-contiguously in memory. Each node contains data and a pointer. Unlike arrays, linked lists do not require a predefined size and can grow or shrink dynamically. Types of linked list include **singly linked list** and **doubly linked list**.*

   a. Laksanakan sebuah program yang menggabungkan dua **senarai terpaut tunggal** yang disusun, $L1$ dan $L2$, menjadi satu senarai terpaut tunggal yang disusun $L3$ dalam turutan menaik. Fungsi tersebut harus berfungsi dalam $O(n)$, di mana n ialah jumlah keseluruhan nod dalam kedua-dua senarai. Penggunaan ruang tambahan mestilah $O(1)$. Penggabungan mesti dilakukan dalam keadaan setempat dengan menyusun semula penunjuk nod dan tiada nod baru yang perlu dicipta.

   *Implement a program that merges two sorted **singly linked lists** $L1$ and $L2$ into a single sorted linked list $L3$ in increasing order. The function should work in $O(n)$, where n is the total number of nodes in both lists. The additional space usage must be $O(1)$. The merge must be done in-place by rearranging the node pointers and no new nodes should be created.*

   Nota: Anda **HARUS** mengubah suai senarai terpaut yang sedia ada secara langsung dan mengembalikan penunjuk kepala bagi senarai terpaut yang digabungkan, $L3$.

   ***Note**: You **MUST** modify the existing linked lists directly and return the head pointer of the merged linked list $L3$.*

   Input: $L1: 1, 10, 20$ and $L2: 15, 25, 55$
   Output:

```
Merged Linked List:
1 -> 10 -> 15 -> 20 -> 25 -> 55 -> null
```

(5 markah/*marks*)

b. Palindrom adalah perkataan, nombor, frasa, atau turutan yang dibaca sama terbalik dan secara langsung. Jika anda membalikkan urutan aksara, ia tetap kelihatan sama!

A palindrome is a word, number, phrase, or sequence that reads the same backward as forward. if you reverse the order of the characters, it still looks exactly the same!

Laksanakan sebuah program untuk menentukan sama ada **senarai terpaut ganda** yang diberikan (dibina daripada sebuah string) adalah palindrom. Penggunaan ruang tambahan mestilah $O(1)$, anda **TIDAK BOLEH** menggunakan struktur data tambahan seperti *array* atau *stack*. Anda hanya boleh mengubah suai penunjuk nod dalam senarai terpaut ganda untuk memeriksa sifat palindrom.

*Implement a program to determine whether a given **doubly linked list** (constructed from a string) is a palindrome. The additional space usage must be $O(1)$, you **MUST NOT** use additional data structures like arrays or stacks. You should only manipulate the node pointers of the doubly linked list to check for palindrome property.*

Nota: Anda boleh menggunakan dua penunjuk, satu bermula dari kepala senarai dan satu lagi dari ekor, gerakkan kedua-dua penunjuk ke arah tengah sambil membandingkan data nod yang ditunjuk, dan berhenti apabila kedua-dua penunjuk bertemu atau melintasi antara satu sama lain. Kembalikan **TRUE** jika senarai adalah palindrom dan **FALSE** jika sebaliknya.

***Note**: You may use two pointers, one starting from the head of the list and one from the tail, move both pointers towards the center while comparing the data of the nodes they point to and stop when the two pointers meet or cross each other. Return **TRUE** if the list is a palindrome and **FALSE** otherwise.*

Input: "rotator"
Output:

```
Input String: rotator
Is Palindrome? true
```

Input: "racecar"
Output:

```
Input String: racecar
Is Palindrome? true
```

Input: "robot"
Output:

```
Input String: robot
Is Palindrome? false
```

(5 markah/*marks*)

2. Laksanakan sebuah **struktur data barisan berpusing** generik dengan menggunakan ***Array***.

Implement a ***circular generic queue*** *data structure using* ***Array***.

a. Barisan berpusing generik harus terdiri daripada kaedah-kaedah berikut:

The circular generic queue should consist of the following methods:

1.	constructor	Pembina lalai untuk baris gilir generik bulat. *Default constructor for circular generic queue.*
2.	enqueue	Menambahkan elemen pada bahagian belakang baris gilir . *Add element to the end of the queue.*
3.	dequeue	Mengeluarkan elemen pada bahagian belakang baris gilir. *Remove element at the end of the queue.*
4.	toString	Paparkan semua elemen dalam baris gilir dalam susunan ke hadapan.. *Display all elements in the queue in forward order.*

Baris gilir generik bulat mempunyai kapasiti maksimum sebanyak 8, dan tiada perubahan saiz dibenarkan. Jika proses *enqueue* melebihi saiz, elemen yang paling lama akan digantikan.

The circular generic queue has a maximum capacity of 8, and no resize is allowed. If the enqueue process exceeds the size, oldest elements will be replaced.

(5 markah/*marks*)

b. Jana secara rawak 10 nombor matrik dengan format "SNNNNNNNN", di mana nombor matrik adalah *String*, bermula dengan satu aksara dan diikuti oleh 8 digit. Masukkan semua nombor matrik ke dalam sebuah baris gilir seperti dalam (a). Cetak semua elemen dalam barisan.

Randomly generate 10 matric numbers with the format of "SNNNNNNNN", where the matric numbers are Strings, start with one character and followed by 8 digits. Insert all the matric numbers by creating a queue as in (a). Print all the elements in the queue.

(3 markah/*marks*)

c. Laksanakan semula (a) dengan menggunakan konsep baris gilir dengan keutamaan (*priority queue*) dengan kekangan kapasiti yang sama, di mana nombor matrik dengan keutamaan tertinggi akan diganti jika melebihi saiz. Program perlu mengutamakan pengisihan berdasarkan aksara pertama bagi nomor matrik secara tertib menaik, dan kemudian mengisih berdasarkan baki digit secara tertib menurun.

Reimplement (a) using the priority queue concept with the same capacity constraints, where matric number with the highest priority will be replaced if size exceeded. The

program should prioritize sorting on the first character of the matric number in ascending order, then only sort based on the rest of the digits in descending order.

Output:

```
Random matric number: P74901557
Random matric number: Z16435641
Random matric number: F45975507
Random matric number: W49879487
Random matric number: J37378713
Random matric number: L96353582
Random matric number: L80558031
Random matric number: H67746522
Random matric number: C90133794
Queue is full! Replaced P74901557
PriorityQueue is full! Replaced F45975507
Random matric number: B35454922
Queue is full! Replaced Z16435641
PriorityQueue is full! Replaced C90133794
Queue: F45975507 W49879487 J37378713 L96353582 L80558031
H67746522 C90133794 B35454922
PriorityQueue: B35454922 H67746522 J37378713 L96353582
L80558031 P74901557 W49879487 Z1643564
```

(7 markah/*marks*)

3. Anda merupakan seorang pegawai daripada Jabatan Teknologi Maklumat di sebuah universiti dan telah ditugaskan untuk membangunkan modul *backend* bagi Sistem Rekod Akademik. Anda diberikan satu fail teks bernama *students.txt*, yang mengandungi **50** rekod pelajar. Setiap rekod terdiri daripada tiga medan yang dipisahkan dengan koma, iaitu **ID Pelajar (integer)**, **Nama (String)**, dan **Skor (double).**

*You are part of a university's IT department tasked with developing a backend module for the Academic Records System. You are provided with a text file named students.txt, containing **50 student records** Each record consists of three fields separated by commas, **Student ID (integer)**, **Name (String)**, and **Score (double)**.*

Baris 10 pertama di dalam *students.txt*:

The first 10 lines in student.txt:

```
103,Carol,91.2
118,Robert,85.6
102,Bob,76.0
115,Olivia,95.0
117,Queenie,70.4
122,Victor,82.9
119,Sophia,89.9
112,Liam,67.0
111,Kate,92.1
113,Mia,80.5
```

Merujuk kelas Student:

Refer Student class:

```
class Student {
    int id;
    String name;
    double score;

    Student(int id, String name, double score) {
        this.id = id;
        this.name = name;
        this.score = score;
```

Anda dikehendaki untuk:

You are required to:

a. Baca Rekod Pelajar daripada Fail

Read Student Records from File

- Baca data daripada students.txt.

  Read data from students.txt.

  Setiap baris mengandungi ID Pelajar (int), Nama (String), dan Skor (double).

  Each line has a Student ID (int), Name (String), and Score (double).

- Cipta objek Student bagi setiap rekod dan simpan dalam ArrayList.

  Create a Student object for each record and store them in an ArrayList.

b. Susun Pelajar mengikut ID Pelajar (Insertion Sort)

Sort Students by Student ID (Insertion Sort)

- Cipta satu kaedah untuk mengisih senarai pelajar berdasarkan ID Pelajar dalam susunan menaik.

  Create a method to sort the list of students based on their Student ID in ascending order.

- Gunakan algoritma Insertion Sort.

  Use Insertion Sort algorithm.

- Pastikan pengisihan menyusun semula pelajar dengan betul.

  Make sure the sorting correctly reorders the students.

c. Cari Pelajar menggunakan Binary Search

Search for a Student using Binary Search

- Cipta satu kaedah untuk melaksanakan Binary Search bagi mencari pelajar berdasarkan ID Pelajar.

  *Create a method to perform a **Binary Search** to find a student by Student ID.*

  Pencarian mesti dilakukan selepas senarai disusun.

  *Search must happen **after** the list is sorted.*

d. Uji Program Anda

Test your program

- Paparkan senarai pelajar yang telah disusun.
  Print sorted student list.
- Cari ID Pelajar: 119
  Search for student ID: 119
- Cari ID Pelajar: 197
  Search for student ID: 197
- Jika pelajar dijumpai, papar maklumat penuh mereka (ID, Nama, Skor).
  If the student is found, display their full information (ID, Name, Score).
- Jika pelajar tidak dijumpai, kembalikan nilai -1.
  If the student is not found, return -1.

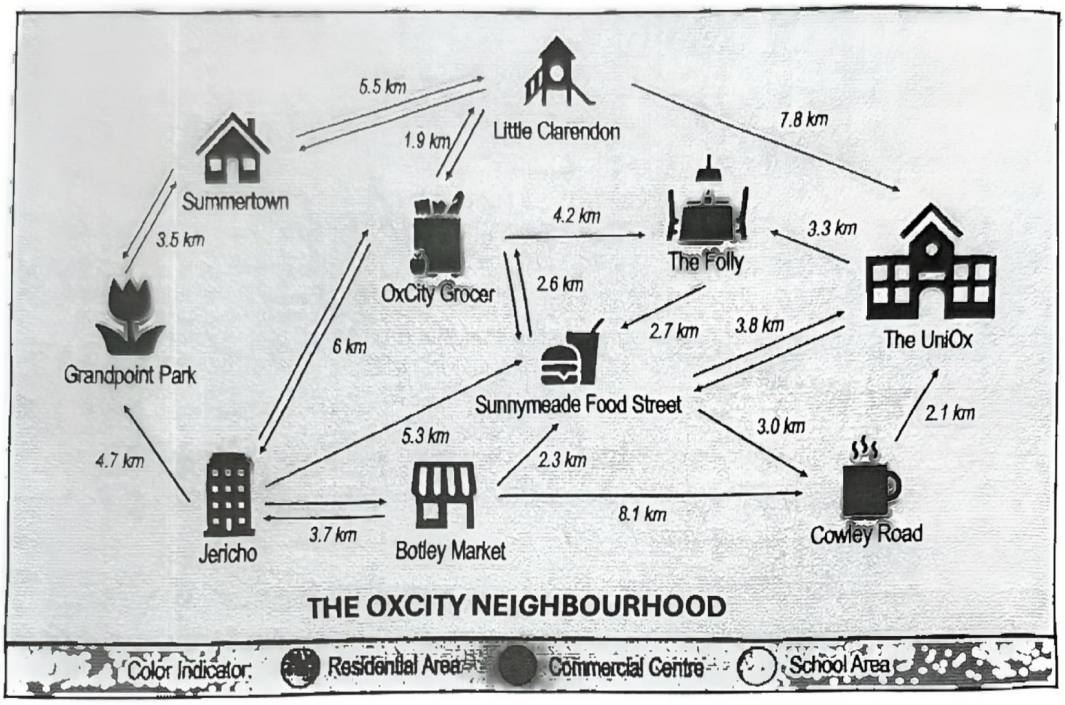
(10 markah/*marks*)

4. "Tahniah! Anda telah diterima masuk sebagai pelajar dalam Program Ijazah Tinggi di UniOx!"

Selepas anda menghabiskan Program Ijazah Dasar di Universiti Malaya, anda telah berhasrat untuk memohon untuk melanjutkan pelajaran di salah satu universiti terbaik di dunia. Hari ini, anda telah menerima surat tawaran daripada universiti tersebut, UniOx! Anda berasa teruja dan tidak sabar hendak mengemas dan berpindah ke kawasan perumahan yang berdekatan dengan universiti. UniOx telah menghantar satu peta seperti yang dipaparkan dalam Rajah 1 untuk anda membiasakan diri dengan OxCity:

"Congratulations! You have been accepted as a Postgraduate Program Student in UniOx!"

After you finished your Undergraduate program at Universiti Malaya, you have decided to apply to further study in one of the best universities in the world. Today, you finally received the offer letter from the university, UniOx! You are excited and can't wait to pack and move to the new neighbourhood near the university. UniOx has sent you a map as shown in Figure 1 for you to get familiarize with the OxCity:



Rajah 1: Sistem penghalaan OxCity.

Figure 1: OxCity routing system.

Anda berancang untuk memandu dalam OxCity ke kampus untuk perjalanan yang mudah. Oleh itu, anda telah mendapatkan anggaran kelajuan memandu daripada pejabat majlis OxCity yang merangkumi pelbagai kawasan:

- Kawasan Perumahan ⇆ Kawasan Perumahan = x min/km
- Kawasan Perumahan ⇆ Pusat Komersial = 1.5x min/km
- Kawasan Perumahan ⇆ Kawasan Sekolah = 1.7x min/km
- Pusat Komersial ⇆ Pusat Komersial = 2.5x min/kim
- Pusat Komersial ⇆ Kawasan Sekolah = 2x min/km
- Kawasan Sekolah ⇆ Kawasan Sekolah = 1.2x min/km

Sebagai contoh, kalau anda memandu daripada Kawasan Perumahan A ke Pusat Komersial B, andaikan x=2, **jarak memandu** adalah 6.2 km dan **tempoh masa**

memandu adalah 18.6 minit kerana **kelajuan memandu** dalam Kawasan Perumahan adalah berkelajuan 1.5x *min/km*.

You are planning to drive in OxCity to the campus for an easy commute. Therefore, you also obtained the estimated driving speed from the OxCity council office within various areas:

- *Residential Area* ⇆ *Residential Area* = x *min/km*
- *Residential Area* ⇆ *Commercial Centre* = 1.5x *min/km*
- *Residential Area* ⇆ *School Area* = 1.7x *min/km*
- *Commercial Centre* ⇆ *Commercial Centre* = 2.5x *min/km*
- *Commercial Centre* ⇆ *School Area* = 2x *min/km*
- *School Area* ⇆ *School Area* = 1.2x *min/km*

For example, if you are commuting from A Residential Area to B Commercial Centre, assuming that x=2, *if the* ***commute distance*** *is 6.2 km and the* ***commute duration*** *will be 18.6 minutes (distance * driving speed) because the* ***driving speed*** *within these areas are of* 1.5x *min/km.*

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

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

a. Bina satu graf untuk menangkap sistem penghalaan dalam bandar OxCity. Khususnya, semua Lokasi dalam Kawasan Perumahan, Pusat Komersial, dan Kawasan Sekolah akan dijadikan *Vertexes* manakala penghalaan dalam lokasi-lokasi dijadikan *Edges*. Dalam kelas **Edge** anda, anda perlukan menangkap kelajuan memandu dan jarak memandu, secara berasingan.

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

   i. `getSize` untuk menunjukkan bilangan *Vertexes*.
   ii. `getVertex` untuk mendapatkan nama lokasi.
   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.

   Selepas itu, bina satu kelas **TestGraph** dan laksanakan cara *main* untuk membina graf dan memaparkan maklumat yang berikut seperti dalam Rajah 2 dengan menggunakan cara-cara utiliti (mengandaikan x=1.5):

Construct a graph to capture the routing system of the OxCity neighbourhood. In particular, all the locations in Residential Area, Commercial Centre, and School Area will be the Vertexes while the routes within these locations will be the Edges. In your **Edge** *class, you should capture the driving speed and the commute distance, separately.*

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 Location 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, construct a **TestGraph** *class and implement a main method to create the graph and display the following information as shown in Figure 2 using the utility methods (assume that* x=1.5*):*

```
Output - (TestGraph.app) ×
The number of vertices in MyCityGraph: 10
List all vertices:
0: Jericho  1: Grandpoint Park  2: Summertown  3: Botley Market  4: Cowley Road  5: Sunnymeade Food Street  6: OxCity Grocer  7: The Folly  8: Little Clarendon  9: The UniOx
Has edge from Little Clarendon to Summertown?  true
Has edge from Summertown to The UniOx?  false

Find all neighbours of OxCity Grocer : [Sunnymeade Food Street, The Folly, Little Clarendon, Jericho]

Print all edges :
# Jericho : [Jericho, Grandpoint Park (speed=1.0, distance=4.7)] [Jericho, OxCity Grocer (speed=1.5, distance=6.0)] [Jericho, Sunnymeade Food Street (speed=1.5, distance=8.3)] [Jericho, Botley Market (speed=1.5, distance=3.1)]
# Grandpoint Park : [Grandpoint Park, Summertown (speed=1.0, distance=3.5)]
# Summertown : [Summertown, Little Clarendon (speed=1.7, distance=5.5)] [Summertown, Grandpoint Park (speed=1.0, distance=3.5)]
# Botley Market : [Botley Market, Cowley Road (speed=2.5, distance=8.1)] [Botley Market, Sunnymeade Food Street (speed=2.5, distance=2.3)] [Botley Market, Jericho (speed=1.5, distance=3.1)]
# Cowley Road : [Cowley Road, The UniOx (speed=2.0, distance=2.1)]
# Sunnymeade Food Street : [Sunnymeade Food Street, The UniOx (speed=2.0, distance=3.8)] [Sunnymeade Food Street, Cowley Road (speed=2.5, distance=3.0)] [Sunnymeade Food Street, OxCity Grocer (speed=2.5, distance=2.6)]
# OxCity Grocer : [OxCity Grocer, Sunnymeade Food Street (speed=2.5, distance=2.6)] [OxCity Grocer, The Folly (speed=2.5, distance=4.2)] [OxCity Grocer, Little Clarendon (speed=2.0, distance=1.9)] [OxCity Grocer, Jericho (speed=1.5, distance=6.0)]
# The Folly : [The Folly, Sunnymeade Food Street (speed=2.5, distance=2.7)]
# Little Clarendon : [Little Clarendon, The UniOx (speed=1.2, distance=7.8)] [Little Clarendon, OxCity Grocer (speed=2.0, distance=1.9)] [Little Clarendon, Summertown (speed=1.7, distance=5.5)]
# The UniOx : [The UniOx, Cowley Road (speed=2.0, distance=2.1)] [The UniOx, Sunnymeade Food Street (speed=2.0, distance=3.8)] [The UniOx, The Folly (speed=2.0, distance=3.3)]
```

Rajah 2: Keluaran untuk soalan 4. a.

Figure 2: Output for question 4. a.

(6 markah/*marks*)

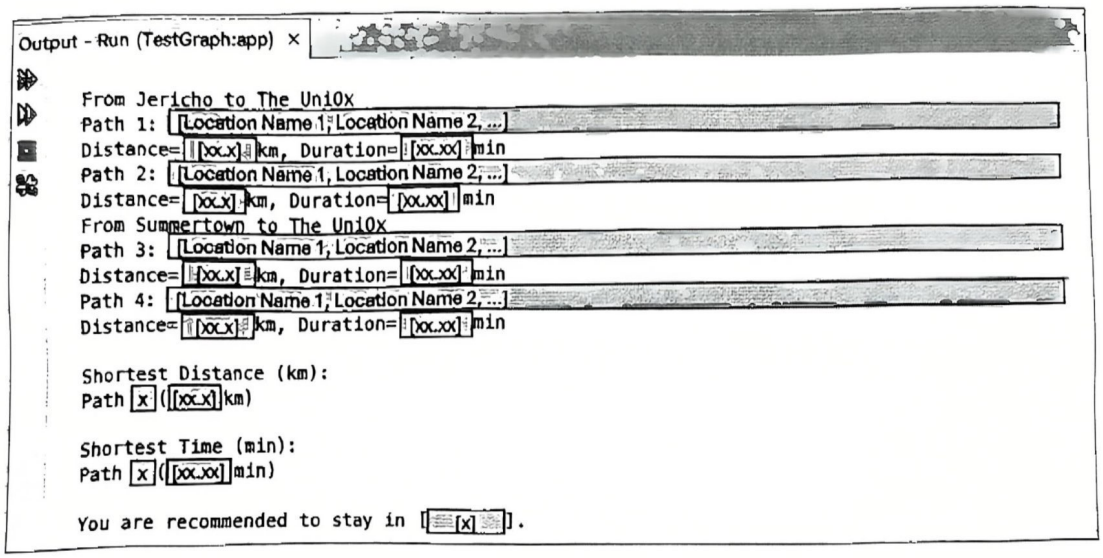
b. Terdapat dua kawasan perumahan yang boleh diduduki dalam bandar OxCity, termasuk Jericho dan Summertown. Senaraikan **DUA (2)** laluan yang boleh digunakan untuk memandu daripada Jericho ke The UniOx dan **DUA (2)** laluan yang boleh digunakan untuk memandu daripada Summertown ke The UniOx. Setiap *Vertex* hanya boleh digunakan satu kali untuk mengelakkan gelung.

Dalam kelas **TestGraph** anda, bina dua cara bernama `calculateDistance` dan `calculateDuration` untuk mengira jumlah

jarak dan tempoh masa untuk keempat-empat laluan yang disenaraikan. Lepas itu, dalam cara *main*, panggil cara-cara tersebut dan tunjukkan keluaran berikut seperti dalam Rajah 3:

There are two Residential Area available to stay in the OxCity neighbourhood, including Jericho and Summertown. List ***TWO (2)*** *possible paths to commute from Jericho to The UniOx and* ***TWO (2)*** *possible paths to commute from Summertown to The UniOx. Each Vertex should only be utilized once to avoid loop.*

In your ***TestGraph*** *class, create two methods namely* `calculateDistance` *method and* `calculateDuration` *method to compute the distances and commute duration for the four paths identified. Then, in the main method, call the methods and show the output as in Figure 3:*



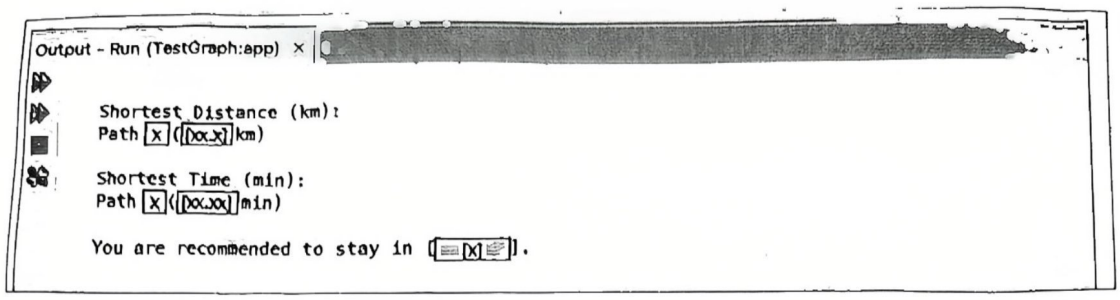
Rajah 3: Keluaran untuk soalan 4. b. Jarak dan tempoh masa anda mungkin berbeza bergantung kepada laluan yang telah disenaraikan.

Figure 3: Output for question 4. b. Your distances, and durations might be different than the shown figure depending on your listed paths.

(6 markah/*marks*)

c. Akhirnya, binakan kod untuk mencari jarak dan masa yang terpendek untuk keempat-empat laluan di soalan 4. b.. Program ini perlu mengesyorkan kawasan perumahan untuk diduduki berdasarkan jarak dan masa yang terpendek.

Finally, construct the code to find out the shortest distance and shortest time for the four identified paths in question 4. b.. The program should give recommendation on which Residential Area to stay based on the shortest distance and shortest time.



Rajah 4: Keluaran untuk soalan 4. c. Jarak, tempoh masa, dan pengesyoran anda mungkin berbeza bergantung kepada laluan yang telah disenaraikan.

Figure 4: Output for question 4. c. Your distances, duration, and recommendation might be different from the shown figure depending on your listed paths.

(3 markah/*marks*)

TAMAT
END