



<b>MODULE NAME:</b>	<b>MODULE CODE:</b>
<b>PROGRAMMING 1B</b>	<b>PROG6112</b>

<b>ASSESSMENT TYPE:</b>	<b>TEST (PAPER ONLY)</b>
<b>TOTAL MARK ALLOCATION:</b>	<b>60 MARKS</b>
<b>TOTAL HOURS:</b>	<b>1 HOUR (+10 minutes reading time)</b>

**SETUP TIME – SPECIAL INSTRUCTIONS:**

1. For practical IT tests or exams written on campus, the usual reading time is replaced by an **additional 30-minute setup time** allocated for setup, saving and upload activities.
2. Students are allowed to make notes during the 30-minute setup time.
3. Students are allowed to **start working** on their practical solutions as soon as the 30-minute setup time starts.

**INSTRUCTIONS:**

1. Please adhere to all instructions in the assessment booklet.
2. Independent work is required.
3. Ten minutes is dedicated to reading time before the start of the assessment. You may make notes on your question paper, but not in your answer sheet. Calculators may not be used during reading time.
4. You may not leave the assessment venue during reading time, or during the first hour or during the last 15 minutes of the assessment.
5. Ensure that your name is on all pieces of paper or books that you will be submitting. Submit all the pages of this assessment's question paper as well as your answer script.
6. Answer all the questions on the answer sheets or in answer booklets provided. The phrase 'END OF PAPER' will appear after the final set question of this assessment.
7. Remember to work at a steady pace so that you are able to complete the assessment within the allocated time. Use the mark allocation as a guideline as to how much time to spend on each section.

**Additional instructions:**

1. This is an OPEN BOOK assessment.
2. Students can use lab computers or their personal devices to connect to Azure Lab Services and to complete the assessment. If students choose to use their own devices they must ensure that they are able to connect to campus networks, Azure Lab Services and the LMS in advance of the assessment sitting. No campus assistance is available during the assessment to troubleshoot problems with personal devices.
3. For open book assessments the students may have open access to all resources inclusive of notes, books (hardcopy and e-books) and the internet. These resources may be accessed as hard copies or as electronic files on electronic devices. All electronic devices batteries must be fully charged before the assessment as no charging of devices will be permitted during the sitting of the assessment. The IIE and its sites of delivery accept no liability for the loss or damage incurred to electronic devices used during open book assessments.
4. Answer All Questions.
5. Instructions for submitting your assessment:

- *Use of good programming practice and comments in code is compulsory.*
- *Save your solution/project in the designated space for this module.*
- *Save all files (including any source code files, template files, design files, image files, text files, database files, etc.) within the designated space.*
- *Do NOT save zipped (archive) files in the designated space unless specifically instructed to do so.*
- ***Important:*** *Upon completion of your assessment, you must save and close all your open files before submitting your work. You will submit your assessment on the LMS page for this module.*
- ***To complete your submission:*** *Create a document in MS Word or Notepad. The document name must follow the format shown here:*
- ***StudentNumber\_ModuleCode\_Test.*** *E.g., if your student number is 12345 and you are writing a Test for the module PROG6112, create a document named **12345\_PROG6112\_Test.***
- *In this document include the following: Your student number, the module code, and **the link to designated space** where you saved your practical work. If you are required to include any written answers, also include type these answers in the same document.*
- *Submit this document in the LMS, using the Practicum submission link for this module.*

**NOTE:**

1. ***Each question indicates a sample result. This should be used as a guide to the expected format and/or result, providing you with a clear understanding of what is expected. However, in some cases, the sample result may only include a single record of the expected results.***
2. ***For mark allocation, it is crucial that you refer to the Marking Rubrics at the end of each question. These rubrics provide a clear breakdown of how your work will be assessed.***

**Question 1****(Marks: 30)**

Develop a Java application with the purpose of recording and displaying road accidents for cars and motorbikes in different cities. The application will prompt the user to input car and bike accidents in a specific city and then display the total number of recorded road accidents that occurred in three different cities. The resulting report will show road accidents for cars and motorbikes, along with the town and the number of accidents for each vehicle type. The table below represents the road accident data for the report generation:

	Car	Motor Bike
Cape Town	155	121
Johannesburg	178	145
Port Elizabeth	112	89

***Sample screenshot***

```
Enter the number of car accidents for Cape Town: 155
Enter the number of motor bike accidents for Cape Town: 121
Enter the number of car accidents for Johannesburg: 178
Enter the number of motor bike accidents for Johannesburg: 145
Enter the number of car accidents for Port Elizabeth: 112
Enter the number of motor bike accidents for Port Elizabeth: 89
```

---

#### ROAD ACCIDENT REPORT

---

	CAR	MOTOR BIKE
Cape Town	155	121
Johannesburg	178	145
Port Elizabeth	112	89

---

#### ROAD ACCIDENT TOTALS FOR EACH CITY

---

Cape Town	276
Johannesburg	323
Port Elizabeth	201

---

CITY WITH THE MOST VEHICLE ACCIDENTS: Johannesburg

---

Create a comprehensive road accident report by utilising single and two-dimensional arrays to accurately capture and analyse the total number of road accidents in each city.

In your solution, also include the city with the highest number of road accidents.

Question 1 Mark Allocation	Levels of Achievement				Feedback
	Excellent	Good	Developing	Poor	
	Score Ranges Per Level (½ marks possible)				
Declaration and Population of single and two-dimensional arrays.	10	5-9	1-4	0	
	Single and two-dimensional arrays were declared and populated without any errors.	Single and two-dimensional arrays were declared and populated with one or two minor errors.	Single and two-dimensional arrays were not declared or declared incorrectly, and the populated arrays had many errors.	Not provided.	
Output: Printing of rows and columns in the report.	5	3-4	1-2	0	
	The report displays or prints all the rows and columns of the report.	The report displays or prints some rows and columns of the report (not all).	The report has major errors and little output of rows and columns.	There is no report of rows and columns.	
Printing and calculating the total road accidents for each city.	10	5-9	1-4	0	
	The calculation and printing of the total road accidents have been implemented correctly.	Minor changes are required.	Major changes are required.	Not provided.	
Determine and display the city with the greatest number of road accidents.	5	3-4	1-2	0	
	Logic was created to determine the city with the greatest number of road accidents.	Minor changes are required.	Major changes are required.	No Output	

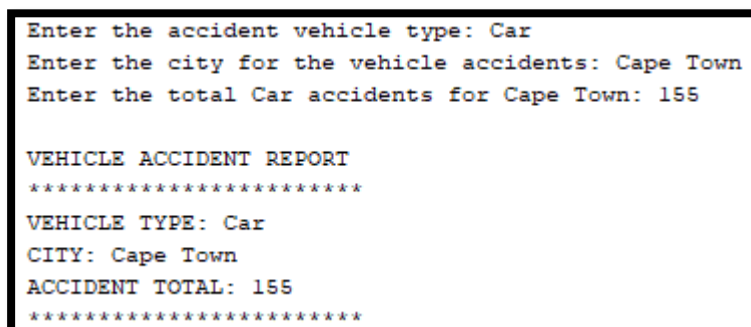
**Question 2****(Marks: 30)**

Design a console application that will print the vehicle accidents report for a city. Make use of an abstract class named RoadAccidents that contains variables to store the vehicle type, the city where the accident occurred and the total number of accidents. Create a constructor that accepts the vehicle type, city, and number of accidents as parameters. In this class also create get methods to get the vehicle type, city, and number of accidents. The RoadAccidents class must implement an IRoadAccidents interface that contains the following:

```
public interface IRoadAccidents
{
    String getAccidentVehicleType();
    String getCity();
    int getAccidentTotal();
}
```

Create a subclass called RoadAccidentReport that extends the RoadAccidents class. The RoadAccidentReport class must contain a constructor to accept the vehicle type, city, and number of accidents as parameters. Write code for the printAccidentReport method, which prints the vehicle type, city and the number of accidents that occurred.

Finally, write a RunApplication class to instantiate the RoadAccidentReport class. Sample output is shown below, and you may use the same values to test your application.

**Sample screenshot**

```
Enter the accident vehicle type: Car
Enter the city for the vehicle accidents: Cape Town
Enter the total Car accidents for Cape Town: 155

VEHICLE ACCIDENT REPORT
*****
VEHICLE TYPE: Car
CITY: Cape Town
ACCIDENT TOTAL: 155
*****
```

Question 2 Mark Allocation	Levels of Achievement				Feedback
	Excellent	Good	Developing	Poor	
	Score Ranges Per Level (½ marks possible)				
IRoadAccidents interface class created	5	3-4	1-2	0	
	IRoadAccidents interface class created correctly.	Minor changes are required.	Major changes are required.	Not provided.	
Abstract Road Accidents class created with a Constructor, Variables and Methods.	5	3-4	1-2	0	
	Abstract Road Accidents class created with a Constructor and the correct Variables and Methods.	Minor changes are required.	Major changes are required.	Not provided.	
Road Accident Report class was created that extends the Road Accidents class and contains a Constructor and the Print Accident Report method.	10	5-9	1-4	0	
	Road Accident Report class was created that extends the Road Accidents class and contains a Constructor and the Print Accident Report method.	Minor changes are required.	Major changes are required.	Not provided.	
The report produced as per sample	5	3-4	1-2	0	
	Report produced correctly	Minor changes are required.	Major changes are required.	No Output	
File saved correctly with suitable comments used in the solution.	5	3-4	1-2	0	
	File has been saved correctly and suitable comments have been used.	Minor changes are required.	Major changes are required.	No Output	

END OF PAPER