A Level 9618

TOPICAL

Computer Science

Paper 2

All Topical | All Variants | Mark Scheme

According to New CAIE 2023-2025 Syllabus

2015-2022

SYED HASEEB BARI

+923351400368

BSS JT, BSS Garden Town

Roots International, Crescent

Book Title: A Level Computer Science Topical Paper 2
Compiled By: Syed Haseeb Bari Gilani (+923351400368)
Published by: STUDENTS RESUORCE® Airport Road Lahore
Edition: 2022-24
Price: 4400/-

NOTICE (warning)

NO FURTHER NOTICE (Legal Warning) WOULD BE ISSUED FOR THIS PURPOSE.

A CATALOGUE RECORD FOR THIS PUBLICATION IS SUBMITTED AND AVAILABLE in IPO (Intellectual Property Organization of Pakistan)

Hey students!

This topical provides all my wonderful students with all chapter-wise arranged questions for A-Levels Computer Science (9618) Paper 2. I have compiled each and every question according to the latest syllabus which is being used from 2021 session onwards.

The 2021-2023 syllabus had some major changes and updates. The Computer Science (9608) was updated to Computer Science (9618) with new subject code. Some new subtopics were added. However, some of the new syllabus additions were taken from previously taught Paper 4 of A-Levels Computer Science (9608).

Therefore, I spent hours gathering questions from Paper 4 of Computer Science (9608) so the questions of every new subtopic could be collected at one place. All the questions and their answers have been thoroughly matched and checked to confirm that they meet the exact requirements of the updated syllabus of Computer Science (9618).

I have been teaching this subject for around 8 years now in well-known institutions like LGS, BSS, BTSC, Kaizen, International School Lahore, The Lahore Lyceum, Roots Millennium and Roots International. With a legacy of 2 Distinctions of my ex-students, the results of my previous and current students speak for themselves and prove my dedication and contribution in this field.

If you were facing problems regarding what questions to expect and where to find the practice questions for the newly added subtopics then don't worry anymore, it has all been put in one place for all my wonderful students so they can practice and remember that practice makes perfect!

I've specifically written some **important instructions** for my students on the next four pages so carefully & thoroughly go through them before starting your practice!

If you still have any concerns, feel free to contact me through the following mediums:

 **0335-1400368**  **@gilanihaseeb**

 **Syed Haseeb Bari Gilani CS/IT – O/A Level**

I hope all of you will get perfect grades that you aim for and undoubtedly distinctions too. Good luck champions!

With lots of wishes for your success,

Haseeb Gilani.

Important Instructions

(regarding Computer Science (9618) syllabus update)

(a) Subject Code (9618):

- The Cambridge has updated the subject code of Computer Science from (9608) to (9618) in the Year 2021.
- This has caused great confusion among the students regarding the syllabus changes that took place.
- It has also caused confusion regarding practice material for the updated subject as there are only a handful of past papers (from 2021 onwards) available for Computer Science (9618).

I hope the following bullet points address all your queries and confusions regarding this update:

- The learning of programming languages for writing **program codes is no longer needed for Paper 2**.
- You will only be required to **write a Pseudocode** for any question asked.
- The **insert booklet** has been introduced which contains all the Pseudocode functions & operators.
- A **few new topics have been added** to the syllabus of the updated 9618 subject which are discussed in **Instruction (b)** on the next page.
- The **rest of the syllabus is completely same** and therefore practicing past paper questions of Computer Science (9608) is totally valid and acceptable.
- Therefore, to prepare for the Computer Science (9618) examination, you must **practice all the past paper questions of Computer Science (9608)** in addition to **practicing newly added topics of Computer Science (9618)** whose questions have been provided in this topical.

(b) Newly Added Topics:

- A few new topics have been added to the Computer Science (9618) – Paper 2.
- These topics have in fact been moved from the previous:

  Computer Science (9608) – Paper 4 to the updated **Computer Science (9618) – Paper 2**

The new syllabus additions have been **bolded** throughout the topical for classification but for further understanding and clarity of the students, a table has been provided below.

The following table summarizes which specific topics have been added and the specific chapters where you can find them in this topical:

Chapter in Topical	Topic Added
Data Types & Records	Record Structure
Abstract Data Types (ADT)	Stack, Queue & Linked List
Structure Charts & State-Transition Diagrams	State-Transition Diagrams
Errors, Testing Methods, Test Data & Maintenance	Testing Methods, Test Strategy/Plan & Test Data

(c) Program Code:

- Writing a **Program Code is no longer** a part of the 9618 syllabus.
- You will only be required to **write a Pseudocode** for any statement, declaration, assignment, selection, iteration, algorithm, procedure, function, array etc.

However, when practicing questions of 9608 syllabus from the topical, you will notice that almost every other question requires you to write a program code as per 9608 syllabus.

Those questions are typically in the following format:

Write **program code** for the function `Search()`.

Visual Basic and Pascal: You should include the declaration statements for variables.
Python: You should show a comment statement for each variable used with its data type.

Programming language .....

Program code

.....

- You must write a **Pseudocode** for practicing such questions and then refer to the mark scheme for checking their answers/solutions.
- The mark schemes only contain Pseudocode answers for these questions, so you do not have to stress about the program code part.
- Therefore, whenever a program code is asked in a question, just write a Pseudocode for it instead and then check your answers/solutions from the mark scheme for practicing.

(d) Appendix (now replaced with 'Insert'):

- The appendix is a list of **built-in Pseudocode functions & operators**.
- The appendix used to be given on the last page(s) of the 9608 examination paper.

When practicing questions of 9608 syllabus from the topical, you will notice that some questions require you to:

'Refer to the **Appendix**'

Whenever you encounter such questions, you will find the appendix with the functions/operators required for that specific question on:

- either the same page
- or next page
- or previous page
- or within a range of 2-10 pages of that question.

That appendix would be taken from the particular paper of which that specific question belongs so it will contain all the desired functions/operators needed.

(e) Insert:

- The term previously used for list of built-in Pseudocode functions & operators called **'Appendix'** has now been replaced with **'Insert'**.
- The insert is a **4 pages booklet** given along with Computer Science 9618 examination paper.
- It is a complete list of all **Pseudocode functions & operators**.
- Therefore, when attempting questions, you can simply open the insert booklet alongside your paper booklet and refer to it simultaneously for using the built-in functions & operators.

When practicing questions of 9618 syllabus from the topical, you will notice that some questions require you to:

'Refer to the **Insert**'

Those questions are typically in the following format:

(a) Write pseudocode for module `RandomChar()`.

You may wish to refer to the **insert** for a description of the `CHR()` function. Other functions may also be required.

- You must use the **Insert** for answering such questions.

While not every question may ask you to 'Refer to the insert':

- Therefore, you will also have to use your own brain, analyze the question, make judgements and then **use the insert yourself accordingly** for the built-in functions & operators even if it is not mentioned in the question.

Table of Contents

Insert – Paper 2

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2022**

INSERT **2 hours**

INFORMATION

- This insert contains all the resources referred to in the questions.
- You may annotate this insert and use the blank spaces for planning. **Do not write your answers** on the insert.

This document has **4** pages.

Functions

Note: an error will be generated if a function call is not properly formed or if the parameters are of an incorrect type or an incorrect value.

String and Character Functions

`LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING`
returns leftmost `x` characters from `ThisString`
Example: `LEFT("ABCDEFGH", 3)` returns `"ABC"`

`RIGHT(ThisString : STRING, x : INTEGER) RETURNS STRING`
returns rightmost `x` characters from `ThisString`
Example: `RIGHT("ABCDEFGH", 3)` returns `"FGH"`

`MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING`
returns a string of length `y` starting at position `x` from `ThisString`
Example: `MID("ABCDEFGH", 2, 3)` returns `"BCD"`

`LENGTH(ThisString : STRING) RETURNS INTEGER`
returns the integer value representing the length of `ThisString`
Example: `LENGTH("Happy Days")` returns `10`

`LCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character representing the lower-case equivalent of `ThisChar`
Non upper-case alphabetic characters are returned unchanged.
Example: `LCASE('W')` returns `'w'`

`UCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character representing the upper-case equivalent of `ThisChar`
Non lower-case alphabetic characters are returned unchanged.
Example: `UCASE('a')` returns `'A'`

`TO_UPPER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to upper case.
Example: `TO_UPPER("Error 803")` returns `"ERROR 803"`

`TO_LOWER(ThisString : STRING) RETURNS STRING`
returns a string formed by converting all characters of `ThisString` to lower case.
Example: `TO_LOWER("JIM 803")` returns `"jim 803"`

`NUM_TO_STR(x : <datatype1>) RETURNS <datatype2>`
returns a string representation of a numeric value.
Note: `<datatype1>` may be `REAL` or `INTEGER`, `<datatype2>` may be `CHAR` or `STRING`
Example: `NUM_TO_STR(87.5)` returns `"87.5"`

`STR_TO_NUM(x : <datatype1>) RETURNS <datatype2>`
returns a numeric representation of a string.
Note: `<datatype1>` may be `CHAR` or `STRING`, `<datatype2>` may be `REAL` or `INTEGER`
Example: `STR_TO_NUM("23.45")` returns `23.45`

`IS_NUM(ThisString : <datatype>) RETURNS BOOLEAN`
returns the value `TRUE` if `ThisString` represents a valid numeric value.
Note: `<datatype>` may be `CHAR` or `STRING`
Example: `IS_NUM("-12.36")` returns `TRUE`

`ASC(ThisChar : CHAR) RETURNS INTEGER`
returns an integer value (the ASCII value) of character `ThisChar`
Example: `ASC('A')` returns `65`, `ASC('B')` returns `66` etc.

`CHR(x : INTEGER) RETURNS CHAR`
returns the character whose integer value (the ASCII value) is `x`
Example: `CHR(65)` returns `'A'`, `CHR(66)` returns `'B'` etc.

Numeric Functions

`INT(x : REAL) RETURNS INTEGER`
returns the integer part of `x`
Example: `INT(27.5415)` returns `27`

`RAND(x : INTEGER) RETURNS REAL`
returns a real number in the range `0` to `x` (**not** inclusive of `x`).
Example: `RAND(87)` could return `35.43`

Date Functions

Note: date format is assumed to be DD/MM/YYYY unless otherwise stated.

`DAY(ThisDate : DATE) RETURNS INTEGER`
returns the current day number from `ThisDate`
Example: `DAY(04/10/2003)` returns `4`

`MONTH(ThisDate : DATE) RETURNS INTEGER`
returns the current month number from `ThisDate`
Example: `MONTH(04/10/2003)` returns `10`

`YEAR(ThisDate : DATE) RETURNS INTEGER`
returns the current year number from `ThisDate`
Example: `YEAR(04/10/2003)` returns `2003`

`DAYINDEX(ThisDate : DATE) RETURNS INTEGER`
returns the current day index number from `ThisDate` where Sunday = 1, Monday = 2 etc.
Example: `DAYINDEX(12/05/2020)` returns `3`

`SETDATE(Day, Month, Year : INTEGER) RETURNS DATE`
returns a variable of type `DATE` with the value of `<Day>/<Month>/<Year>`
Example: `SETDATE(26, 10, 2003)` returns a date corresponding to 26 October 2003

`TODAY() RETURNS DATE`
returns a variable of type `DATE` corresponding to the current date.

Text File Functions

`EOF(FileName : STRING) RETURNS BOOLEAN`
returns `TRUE` if there are no more lines to be read from file `FileName`
Note: the function will generate an error if the file is not already open in `READ` mode.

Operators

Note: an error will be generated if an operator is used with a value or values of an incorrect type.

`&`	concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` evaluates to `"Summer Pudding"` Note: may also be used to concatenate a `CHAR` with a `STRING`
`AND`	performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` evaluates to `FALSE`
`OR`	performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` evaluates to `TRUE`
`NOT`	performs a logical `NOT` on a Boolean value Example: `NOT TRUE` evaluates to `FALSE`
`MOD`	finds the remainder when one number is divided by another Example: `10 MOD 3` evaluates to `1`
`DIV`	finds the quotient when one number is divided by another Example: `10 DIV 3` evaluates to `3`

Comparison Operators

`=`
`>`
`<`
`>=`
`<=`
`<>`
	used to compare two items of the same type
	returns `TRUE` if the condition is true, otherwise returns `FALSE`
	
	Notes:
	• may be used to compare types `REAL` and `INTEGER`
	• may be used to compare types `CHAR` and `STRING`
	• case sensitive when used to compare types `CHAR` or `STRING`
	• cannot be used to compare two records
	
	Examples:
	• `"Program" = "program"` evaluates to `FALSE`
	• `Count = 4` evaluates to `TRUE` when variable `Count` contains the value `4`

Questions

Abstraction & Decomposition

Understanding, realizing & analyzing your mistakes is the key to improvement. Keep a track of your mistakes and note down your weak concepts, topics & sub-topics so that you can work extra hard in those areas and gradually achieve perfection in all topics.

Tracking your mistakes & improving them is the ultimate tool to strengthening your weak concepts & turning them into your strongest ones.

Number of Total Questions	
Number of Correctly Attempted Questions	
Number of Wrongly Attempted Questions	

Fill this table at the end after you have practiced all the given questions:

#	Topic/Subtopic/Mistake	Lessons/Guidelines

#	Topic/Subtopic/Mistake	Lessons/Guidelines

Q1. [9608/MJ/16/21]

(a) Structured programming involves the breaking down of a problem into modules.

Give **two** reasons why this is done.

1 .....

.....

2 .....

.....

[2]

Q2. [9608/MJ/19/21]

(a) A program, `CDOrganiser`, will be written to manage the stored information. The program will consist of three modules: `AddCD`, `FindCD` and `RemoveCD`.

Give **three** reasons why it is good practice to construct the program using modules.

1 .....

2 .....

3 .....

[3]

Q3. [9608/MJ/20/22]

(b) Six program modules implement part of an online shopping program. The following table gives the modules and a brief description of each module:

Module	Description
`Shop()`	Allows the user to choose a delivery slot, select items to be added to the basket and finally check out
`ChooseSlot()`	Allows the user to select a delivery time. Returns a delivery slot number
`FillBasket()`	Allows the user to select items and add them to the basket
`Checkout()`	Completes the order by allowing the user to pay for the items. Returns a Boolean value to indicate whether or not payment was successful
`Search()`	Allows the user to search for a specific item. Returns an item reference
`Add()`	Adds an item to the basket. Takes an item reference and a quantity as parameters

(i) The online shopping program has been split into sub-tasks as part of the design process.

Explain the advantages of decomposing the program into modules. Your explanation should refer to the scenario and modules described in **part (b)**.

.....

.....

.....

.....

.....

..... [3]

Q4. [9608/MJ/20/23]

(c) Explain the process of **problem decomposition**. State one reason it may be used.

Explanation .....

.....

.....

Reason .....

.....

.....

[2]

Q5. [9608/ON/20/22]

(c) A problem may be decomposed into sub-tasks when designing an algorithm.

Give **three** benefits of using sub-tasks.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

Q6. [9608/ON/20/23]

(d) Consider the following pseudocode:

```
10 DECLARE VarA : INTEGER
11 VarA ← 20
12
13 CALL ProcA(VarA)
14 OUTPUT VarA       // first value output
15
16 CALL ProcB(VarA)
17 OUTPUT VarA       // second value output
18
19
20 PROCEDURE ProcA(BYVALUE ThisValue : INTEGER)
21    ThisValue ← ThisValue + 5
22 ENDPROCEDURE
23
24 PROCEDURE ProcB(BYREF ThisValue : INTEGER)
25    ThisValue ← ThisValue + 5
26 ENDPROCEDURE
```

(e) The procedures `ProcA` and `ProcB` in **part (d)** are examples of program modules.

Give **two** advantages of using program modules in program design.

1 .....

.....

2 .....

.....

[2]

Q7. [9608/MJ/21/23]

(a) The process of decomposition is often applied to a programming problem.

Describe the process of decomposition.

.....

.....

.....

.....

.....

..... [2]

Q8. [9608/ON/21/22]

(a) Describe the term **decomposition** when used to develop algorithms.

.....

.....

.....

.....

.....

..... [3]

Latest Questions from Computer Science (9618)

(according to updated Syllabus)

Q9. [9618/MJ/21/22]

(b) A system is being developed to help manage book loans in a library.

Registered users may borrow books from the library for a period of time.

(i) State **three** items of data that must be stored for each loan.

1 .....

2 .....

3 .....

[2]

(ii) State **one** item of data that will be required in the library system but does not need to be stored for each loan.

..... [1]

(iii) One operation that manipulates the data stored for each loan, would produce a list of all overdue books.

Identify **two other** operations.

Operation 1 .....

.....

Operation 2 .....

.....

[2]

Q10. [9618/ON/21/21]

(c) An airline wants to provide passengers with information about individual flights and allow them to book their flight using an online booking system.

(i) Tick (✓) **one** box in each row of the table to indicate whether each item of information would be essential for the customer when making the booking.

Information	Essential	Not essential
Departure time		
Flight number		
Departure airport		
Aircraft type		
Ticket price		
Number of seats in aircraft		

[3]

(ii) Identify the technique used to filter out information that is not essential when designing the booking system **and** state one benefit of this technique.

Technique .....

Benefit .....

.....

[2]

(iii) Identify **two additional** pieces of essential information that a passenger might need when booking a flight.

1 .....

2 .....

[2]

Q11. [9618/ON/21/22]

(a) A programmer applies decomposition to a problem that she has been asked to solve.

Describe decomposition.

.....

.....

.....

.....

.....

..... [2]

Q12. [9618/MJ/22/21]

The manager of a cinema wants a program to allow users to book seats. The cinema has several screens. Each screen shows a different film.

(a) Decomposition will be used to break the problem down into sub-problems.

Describe **three** program modules that could be used in the design.

Module 1 .....

.....

.....

Module 2 .....

.....

.....

Module 3 .....

.....

.....

[3]

(b) Two types of program modules may be used in the design of the program.

Identify the type of program module that should be used to return a value.

..... [1]

Q13. [9618/MJ/22/22]

(b) The structure chart represents part of a complex problem. The process of decomposition is used to break down the complex problem into sub-problems.

Describe **three** benefits of this approach.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

Q14. [9618/ON/22/21]

A system is being developed to help manage a car hire business. A customer may hire a car for a number of days.

An abstract model needs to be produced.

(a) Explain the process of abstraction **and** state **four** items of data that should be stored each time a car is hired.

Explanation .....

.....

Item 1 .....

Item 2 .....

Item 3 .....

Item 4 .....

[3]

(b) Identify **two** operations that would be required to process the car hire data.

Operation 1 .....

.....

Operation 2 .....

.....

[2]

Q15. [9618/ON/22/23]

A program is being designed for a smartphone to allow users to send money to the charity of their choice.

Decomposition will be used to break the problem down into sub-problems.

Identify **three** program modules that could be used in the design **and** describe their use.

Module 1 .....

Use .....

.....

.....

.....

Module 2 .....

Use .....

.....

.....

.....

Module 3 .....

Use .....

.....

.....

.....

[3]

Marking Scheme

Q1.

4 (a)	• Program code is easier to implement / manage • Modules may be given to different people to develop // given to program specialists • Program code is easier to test / debug / maintain • Encourages the re-usability of program code	Max 2

Q2.

Question	Answer	Marks
6(a)	One mark for each of: 1 To make a more manageable / understandable solution 2 Subroutine may be (independently) tested and debugged 3 Program is easier to maintain	3

Q3.

2(b)(i)	Advantages include: • Easier to solve / implement / program the solution as online shopping is a complex task • Easier to debug / maintain as each module can be tested separately e.g. test `FillBasket()` first then test `Checkout()` • Tasks may be shared among a team of programmer. e.g. `Checkout()` and `Search()` modules could be developed in parallel / by teams with different expertise Note: Must include reference to given scenario to achieve all 3 marks - Max 2 if no reference.	3

Q4.

1(c)	Explanation: Breaking a problem down into sub tasks Reason: Make the problem easier to solve // to make the solution easier to implement / test / maintain	2

Q5.

2(c)	One mark per point, example points: 1 Subtasks make the solution more manageable // make the algorithm easier to follow 2 A subtask makes the problem easier to solve / design / program than the whole task 3 A subtask is useful when a part of the algorithm is repeated	3

Q6.

4(e)	Max 2 marks, example answers: • Allows the module to be called from many / multiple places // re-used • Module code can be (independently) tested and debugged once and can then be used repeatedly • If the module task changes the change needs to be made only once • Reduces unnecessary code duplication • Allows modules to be shared among many programmers / given to programmers with specific skills • Makes the program easier to work on / debug / test / etc	2

Marking Scheme

Q7.

3(a)	• To break the problem down into sub-tasks • where each sub-task can be implemented by a program module / is easier to solve. One mark for each phrase (or equivalent)	**2**

Q8.

Question	Answer	Marks
3(a)	One mark each to max 3 • break the problem/algorithm (not program / code) into smaller steps / parts/ subproblems • ... repeatedly only if MP1 given • … until all subproblems small/detailed enough to solve • … to identify program modules // to identify repeated elements // for modular programming • … to identify subroutines	**3**

Q9.

2(b)(i)	Answers include: • User ID / Username • Book ID • Date of loan / return date One mark for 1 correct Two marks for all 3 correct **Note: Max 2 marks**	**2**
2(b)(ii)	Many examples but must be data that is NOT required for a loan, but which COULD be required somewhere by the library system. Note: must be data relating to users, books or loans Answers include: • Users name / address / phone number / DOB • Book title / author / publisher / library rack number / ISBN number / price • Date of loan / return date (if not already given in part (i)) • The length of the loan (assumed to be the same for all books)	**1**
2(b)(iii)	Many examples including: • Create loan / borrow book • Return book • Send letter / email / contact a user ref an overdue book • View the loan history for a given book • View the loan history for a given user One mark for each **Note: Max 2 marks**	**2**

Marking Scheme

Q10.

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>1(c)(i)</td><td>
<table>
<tr><th>Information</th><th>Essential</th><th>Not essential</th></tr>
<tr><td>Departure time</td><td>✓</td><td></td></tr>
<tr><td>Flight Number</td><td></td><td>✓</td></tr>
<tr><td>Departure airport</td><td>✓</td><td></td></tr>
<tr><td>Aircraft type</td><td></td><td>✓</td></tr>
<tr><td>Ticket price</td><td>✓</td><td></td></tr>
<tr><td>Number of seats in aircraft</td><td></td><td>✓</td></tr>
</table>
One mark for two rows correct
Two mark for four rows correct
Three mark for all rows correct
</td><td>3</td></tr>
<tr><td>1(c)(ii)</td><td>One mark for technique and one for benefit, Max 1 mark for 'Benefit'

Technique: Abstraction

Benefit:
• The solution is simplified so easier / quicker to design / implement
• The system is tailored to the need of the user</td><td>2</td></tr>
<tr><td>1(c)(iii)</td><td>Answers include:

• Destination / arrival airport
• Arrival time / flight duration
• Date of flight
• Seat number
• Seat availability

Max 2 marks</td><td>2</td></tr>
</table>

Q11.

Question	Answer	Marks
1(a)	The process involves: 1 Breaking down a problem / task into sub problems / steps / smaller parts 2 In order to explain / understand // easier to solve the problem 3 Leading to the concept of program modules // assigning problem parts to teams **Max 2**	2

Marking Scheme

Q12.

3(a)	One mark per description of appropriate sub-problem for given scenario. Examples include: • Allows the user to search for films being shown // input name of film they want to see • Allows the user to search for available seats • Calculate cost of booking • Book a given number of seats for a particular screening	**3**
3(b)	Function	**1**

Q13.

3(b)	One mark per point: • Breaking a complex problem down makes it easier to understand / solve // smaller problems are easier to understand / solve • Smaller problems are easier to program / test / maintain • Sub-problems can be given to different teams / programmers with different expertise // can be solved separately	**3**

Q14.

2(a)	One mark for Explanation: • Abstraction is used to filter out information / data that is not necessary for the task Or the opposite: • To keep only information / data that is necessary for the task One mark for each **TWO** data items (not dependent on 'Explanation'): Items include: • Car details: ID, Car Registration, car type etc • Customer details: ID, name, address, licence details etc • Start date (of hire) • Return date / Number of days (of hire) • Cost of hire	**3**
2(b)	One mark for each (**Max 2**) Examples include: 1 Input customer details 2 Input car details 3 Input payment details 4 Create hire / start hire 5 Return car / end hire 6 Change / check car status (hired / available / written-off) 7 Cancel hire 8 Process payment / calculate hire cost	**2**

Marking Scheme

Q15.

Question	Answer	Marks
2	One mark for name **and** two marks for use (**Max 3 in total**): Examples include: Module: `SelectCharity()` Use: Allows the user to choose a particular charity Module: `SpecifyAmountAndType()` Use: Allows the user to specify a single or regular payment Module: `MakePayment()` Use: Make payment to the charity Module: `ValidatePayment()` Use: Validate payment details (by accessing bank computer) Module: `AddBankAccountDetails() / AddPaymentDetails()` Use: Allows the user to add bank account information that donation to be taken from Module: `AddDonorDetails()` Use: Allows user to add details such as name and contact details	**3**

Questions

Algorithms, Pseudocodes & Flowcharts

Understanding, realizing & analyzing your mistakes is the key to improvement. Keep a track of your mistakes and note down your weak concepts, topics & sub-topics so that you can work extra hard in those areas and gradually achieve perfection in all topics.

Tracking your mistakes & improving them is the ultimate tool to strengthening your weak concepts & turning them into your strongest ones.

Number of Total Questions	
Number of Correctly Attempted Questions	
Number of Wrongly Attempted Questions	

Fill this table at the end after you have practiced all the given questions:

#	Topic/Subtopic/Mistake	Lessons/Guidelines

#	Topic/Subtopic/Mistake	Lessons/Guidelines

Q1. [9608/MJ/15/21]

A marathon runner records their time for a race in hours, minutes and seconds.

An algorithm is shown below in structured English.

INPUT race time as hours, minutes and seconds
CALCULATE race time in seconds
STORE race time in seconds
OUTPUT race time in seconds

(a) The identifier table needs to show the variables required to write a program for this algorithm.

Complete the table.

Identifier	Data type	Description
`RaceHours`	`INTEGER`	The hours part of the race time.

[3]

(b) Before the program is written, the design is amended.

The new design includes input of the runner's current personal best marathon time (in seconds).

The output will now also show one of the following messages:

- "Personal best time is unchanged"
- "New personal best time"
- "Equals personal best time"

(i) Show the additional variable needed for the new design.

Identifier	Data type	Description

[1]

(ii) Write **program code** for the new design.

Visual Basic and Pascal: You should include the declaration statements for variables.

Python: You should show a comment statement for each variable used with its data type.

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[7]

(c) The program code will be tested using white-box testing.

(ii) Complete the table heading.

Complete Test Number 1.

Add the data for Test Number 2 and Test Number 3.

	Input values				Output	
Test number	**Race hours**	**Race minutes**	**Race seconds**	**.....**	**Total time (seconds)**	**Message**
1	3	4	13	11053	11053	
2				11053		
3				11053		

[6]

Q2. [9608/MJ/15/23]

Horses are entered for a horse race. A horse may have to carry a penalty weight in addition to the rider. This weight is added to the saddle. The penalty weight (if any) depends on the number of wins the horse has achieved in previous races.

The penalty weight is calculated as follows:

Number of previous wins	Penalty weight (kg)
0	0
1 or 2	4
Over 2	8

A program is to be written from the following structured English design.

```
1    INPUT name of horse
2    INPUT number of previous wins
3    CALCULATE penalty weight
4    STORE penalty weight
5    OUTPUT name of horse, penalty weight
```

(a) Complete the identifier table showing the variables needed to code the program.

Identifier	Data type	Description

[3]

(b) Line 3 in the algorithm above does not give the detail about how the race penalty weight is calculated; this step in the algorithm must be expressed in more detail.

(i) The algorithm above currently has five stages. One technique for program design is to further break down, where required, any stage to a level of detail from which the program code can be written.

Name this technique.

.....[1]

(ii) Write **pseudocode** for the given structured English design.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[5]

Q3. [9608/ON/15/21]

A program is to simulate the operation of a particular type of logic gate.

- The gate has two inputs (TRUE or FALSE) which are entered by the user.
- The program will display the output (TRUE or FALSE) from the gate.

The program uses the following identifiers in the pseudocode below:

Identifier	Data type	Description
InA	BOOLEAN	Input signal
InB	BOOLEAN	Input signal
OutZ	BOOLEAN	Output signal

```
1  INPUT InA
2  INPUT InB
3  IF (InA = FALSE AND InB = FALSE) OR (InA = FALSE AND InB = TRUE)
                                       OR (InA = TRUE AND InB = FALSE)
4     THEN
5        OutZ ← TRUE
6     ELSE
7        OutZ ← FALSE
8  ENDIF
9  OUTPUT OutZ
```

(a) The programmer chooses the following four test cases.

Show the output (OutZ) expected for each test case.

Test case	Input		Output OutZ
	InA	InB	
1	TRUE	TRUE	
2	TRUE	FALSE	
3	FALSE	TRUE	
4	FALSE	FALSE	

[4]

(b) The selection statement (lines `03` – `08`) could have been written with more simplified logic.

Rewrite this section of the algorithm in **pseudocode**.

.....

.....

.....

.....

.....

.....[3]

Q4. [9608/ON/15/21]

A program is to be written to calculate the discount given on purchases.

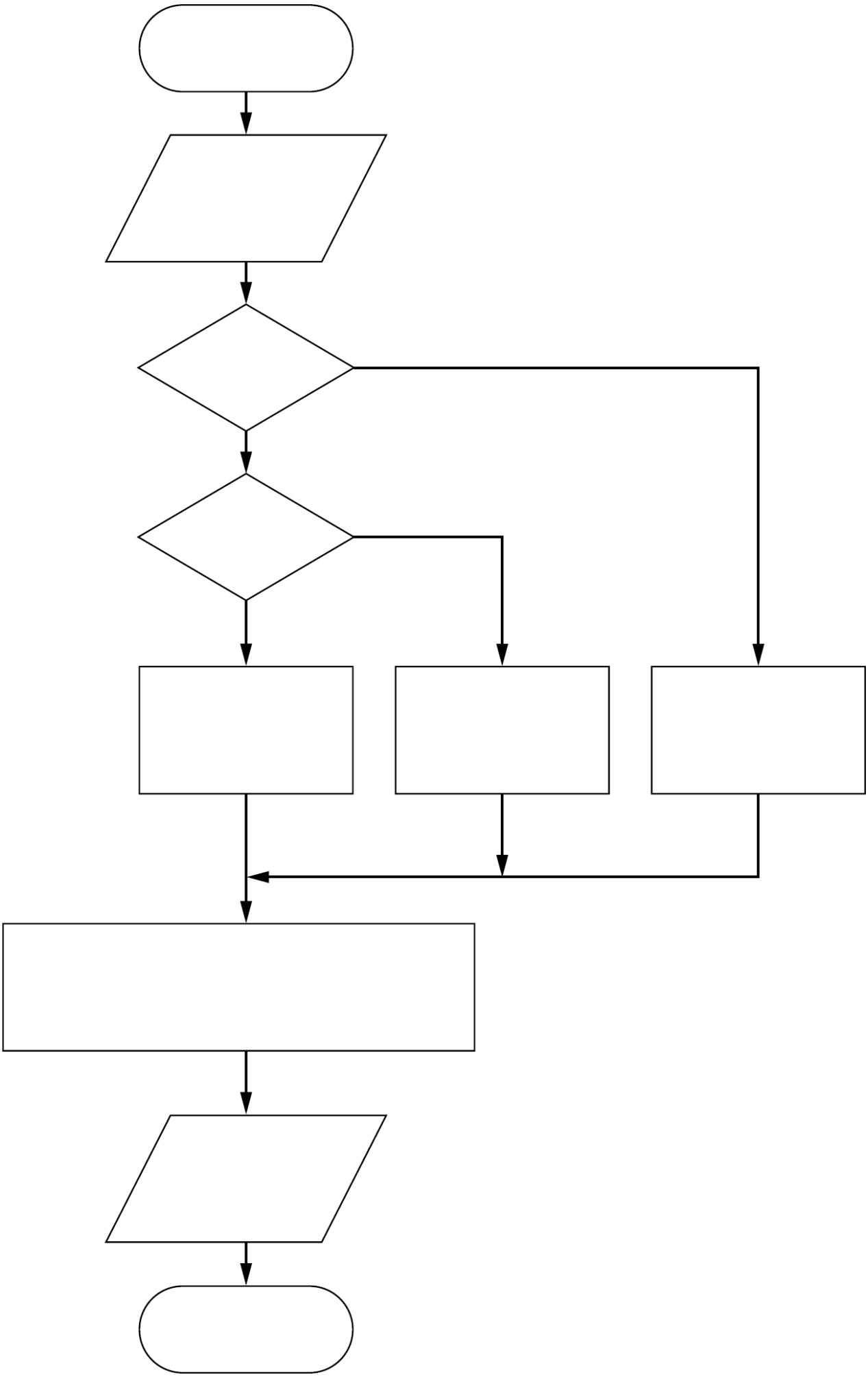
A purchase may qualify for a discount depending on the amount spent. The purchase price (`Purchase`), the discount rate (`DiscountRate`) and amount paid (`Paid`) is calculated as shown in the following pseudocode algorithm.

```
INPUT Purchase

IF Purchase > 1000
   THEN
      DiscountRate ← 0.10
   ELSE
      IF Purchase > 500
         THEN
            DiscountRate ← 0.05
         ELSE
            DiscountRate ← 0
      ENDIF
ENDIF

Paid ← Purchase * (1 - DiscountRate)
OUTPUT Paid
```

The algorithm is also to be documented with a program flowchart.

Complete the flowchart by:

- filling in the flowchart boxes
- labelling, where appropriate, lines of the flowchart



Q5. [9608/ON/15/22]

Regular customers at a supermarket use a rewards card at the point-of-sale.

Points are calculated from every transaction and added to the points total stored on the card.

One reward point is given for every $1 spent.

When the points total exceeds 500, the customer can either:

- pay the full amount due and increase their points total
- get $1 deducted from the amount due in exchange for 500 reward points

The new points total and amount to be paid is printed on the receipt.

A program is to be written with the following specification:

- read the points total from the card
- process the amount spent
- output the amount to be paid and the new points total

A user-defined function `CalculatePoints` has already been coded to calculate the new points earned from the amount spent.

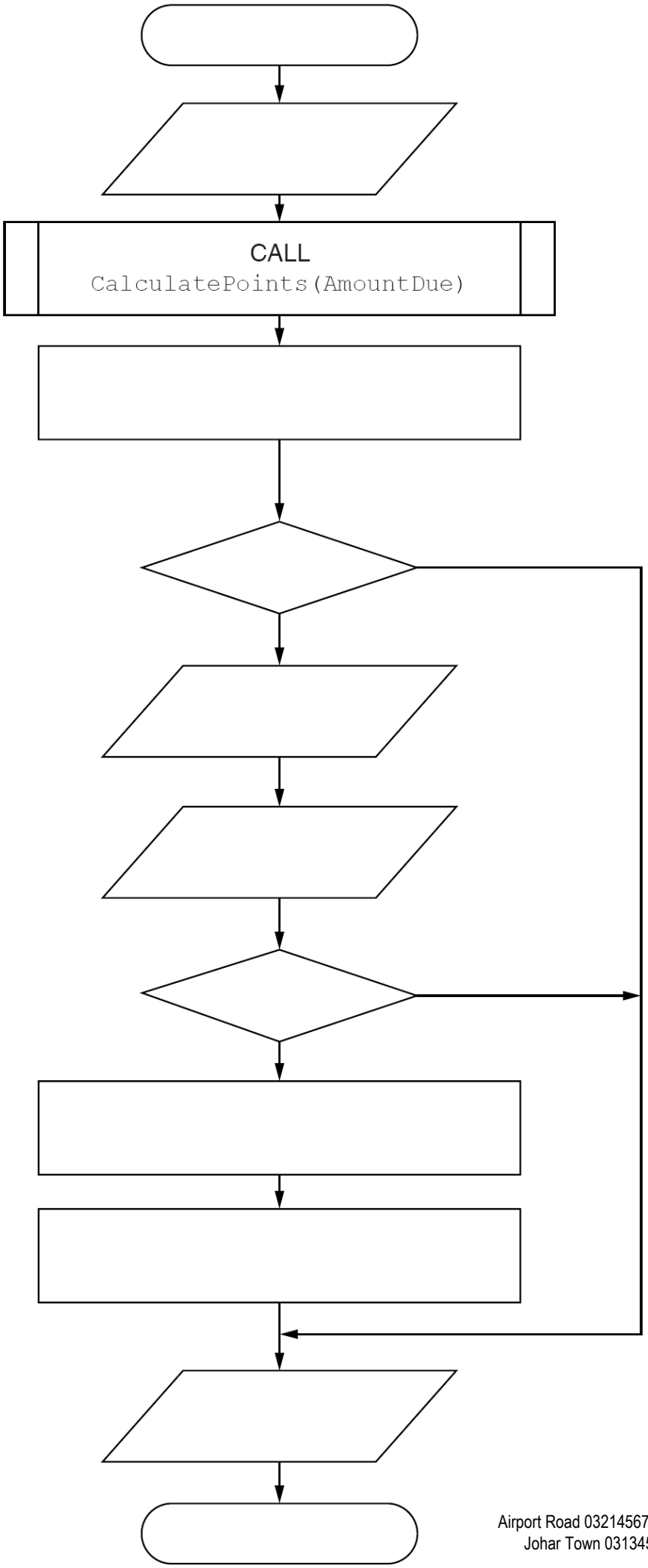
Study the following pseudocode:

```
INPUT AmountDue
NewPoints ← CalculatePoints(AmountDue)
PointsTotal ← PointsTotal + NewPoints

IF PointsTotal > 500
   THEN
      OUTPUT "Exchange points?"
      INPUT Response
      IF Response = "YES"
         THEN
            PointsTotal ← PointsTotal - 500
            AmountDue ← AmountDue - 1
      ENDIF
 ENDIF

 OUTPUT AmountDue, PointsTotal
```

The algorithm is also to be documented with a program flowchart.

Complete the flowchart by:

- filling in the flowchart boxes
- labelling, where appropriate, lines of the flowchart


CALL
CalculatePoints(AmountDue)

Q6. [9608/MJ/16/23]

The engine management system of a car includes an energy-saving facility. When certain conditions are met, this facility will automatically stop the engine.

The system is to be software-based. It will include a subroutine, `EnergySaver`, which repeatedly takes data from sensors in the car. The subroutine decides whether or not to set the `EngineStop` value.

The table of identifiers used by this subroutine is shown below.

(a) Complete the identifier table below by stating the data types.

Identifier	Data type	Description
`Accelerator`	.....	Accelerator pedal position Values: 0 to 100 in steps of 1 Meaning: 0: none (not pressed) 100: maximum (fully pressed)
`EngineTemp`	.....	Engine temperature in °C (-50 to +150 stored to 1 decimal place)
`NormalTemp`	.....	Normal engine temperature in °C Whole number; typical value 90
`Speed`	.....	Road speed of car (in km/hr) Values: 0 to 200 in steps of 1
`EngineStop`	.....	Value used to signal engine must be stopped Possible values: TRUE: stop engine FALSE: run engine

[5]

The condition for stopping the engine is that all three of the following are true:

- Accelerator is not pressed
- Engine temperature is normal or above
- Car speed is zero

The initial design stage will produce a prototype of `EnergySaver`, with a user interface.

The structured English for this is:

1. INPUT value for accelerator pedal position
2. INPUT value for engine temperature
3. INPUT value for normal engine temperature
4. INPUT value for car speed
5. EVALUATE engine stopping condition
6. IF stopping condition satisfied SET engine stop value to TRUE
7. IF stopping condition not satisfied SET engine stop value to FALSE
8. OUTPUT message indicating engine stop value

(b) Write the **pseudocode** equivalent of the structured English. Use the identifiers from the table in **part (a)**.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

Q7. [9608/ON/16/21]

A programmer wants to write a program to calculate the baggage charge for a passenger's airline flight.

Two types of ticket are available for a flight:

- economy class (coded E)
- standard class (coded S)

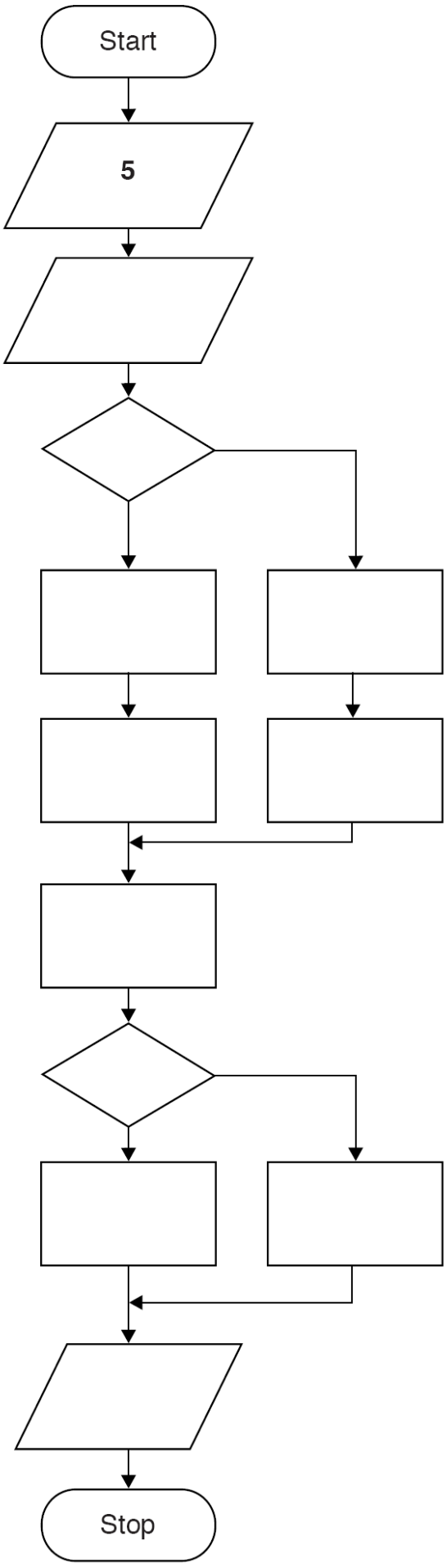
Each ticket type has a baggage weight allowance as shown below. The airline makes a charge if the weight exceeds the allowance.

Ticket type	Baggage allowance (kg)	Charge rate per additional kg ($)
'E'	16	3.50
'S'	20	5.75

(a) A program flowchart will document the program. The flowchart will contain the following statements:

Statement number	Statement
1	`Charge ← 0`
2	`INPUT BaggageWeight`
3	`Charge ← ExcessWeight * ChargeRate`
4	`Is ExcessWeight > 0 ?`
5	`INPUT TicketType`
6	`ExcessWeight ← BaggageWeight - BaggageAllowance`
7	`BaggageAllowance ← 16`
8	`ChargeRate ← 3.5`
9	`OUTPUT Charge`
10	`ChargeRate ← 5.75`
11	`BaggageAllowance ← 20`
12	`Is TicketType = 'E' ?`

Complete the flowchart by putting the appropriate **statement number** in each flowchart symbol. Statement 5 has been done for you.



[6]

Q8. [9608/ON/16/21]

Study the following pseudocode statements.

```
CONST  Pi =  3.1          : REAL

DECLARE Triangle, Base, Height, Radius, Cone : REAL

DECLARE a, b, c, Answer2 : INTEGER

DECLARE Answer1          : BOOLEAN

Base ← 2.6

Height ← 10

Triangle ← (Base * Height) / 2

Radius ← 1

Height ← 2

Cone  ← 2 * Pi * Radius * (Radius + Height)

a ← 13

b ← 7

c ← 3


Answer1 ← NOT((a + b + c) > 28)

Total ← 34

Total ← Total - 2

Answer2 ← a + c * c
```

Give the final value assigned to each variable.

(i) Triangle ..... [1]

(ii) Cone ..... [1]

(iii) Answer1 ..... [1]

(iv) Total ..... [1]

(v) Answer2 ..... [1]

Q9. [9608/ON/16/22]

A number of players take part in a competition. The competition consists of a number of games. Each game is between two players. The outcome of a game is that each player is awarded a grade (A, B, C or D). Each grade has an associated number of points as shown in the table below.

Grade	Points
A	0
B	1
C	3
D	5

The points total for all players is recorded. After each game is completed, the total number of points for each player is updated.

For example:

- before the game between Ryan and Karina, Ryan's total is 5 points and Karina's total is 3 points
- the result of the game between Ryan and Karina is: Ryan achieved grade B, Karina achieved grade D
- the players' points totals are updated to: Ryan has 6 and Karina has 8

When a player's points total reaches 12 or higher, that player is removed from the competition.

A programmer will write a program to update the player total after a game.

The program will output:

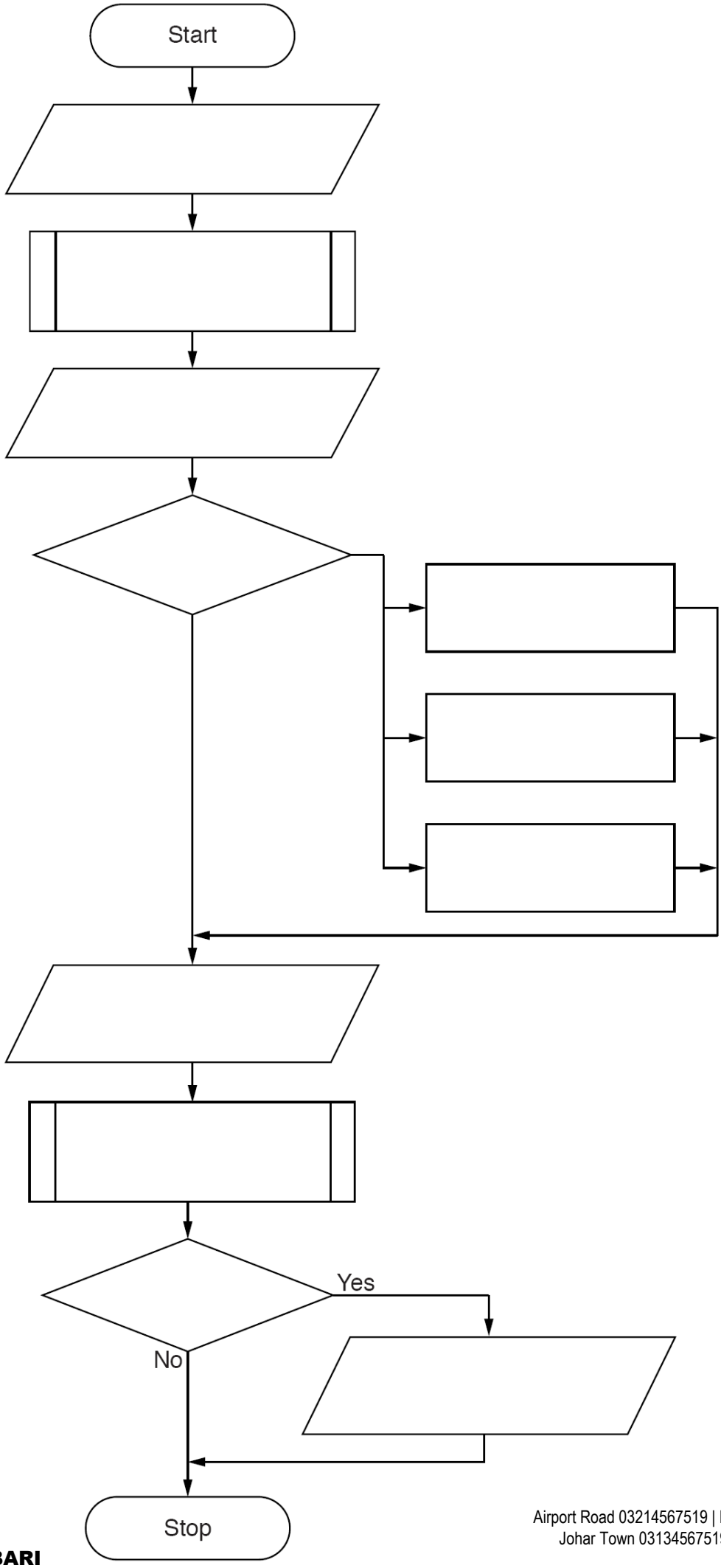
- the player's updated points total
- the message 'ELIMINATED' if the player is removed from the competition.

The programmer designs the identifier table below:

Identifier	Data type	Description
`PlayerName`	`STRING`	Name of the player
`PlayerGameGrade`	`CHAR`	Game grade for the player
`PointsTotal`	`INTEGER`	Current player points
`SavePlayerTotal`	procedure	Procedure has parameters `PlayerName` and `PointsTotal` and saves the updated player total
`ReadPlayerTotal`	function	Function has a parameter `PlayerName` and returns the current total for that player

(a) Complete the following program flowchart by:

- filling in the boxes, using pseudocode where appropriate
- labelling the lines of the flowchart, where necessary. [9]


Start
Yes
No
Stop

(b) Test data is to be produced to test the flowchart.

Complete the table of test data below to show **five** tests that should be used to test different paths through the flowchart.

Test data		Expected results	
`PointsTotal`	`PlayerGameGrade`	Updated `PointsTotal`	Output

[5]