Past-paper walk-through

Data Types and Records ▪ 10.1

eXercise

Questions in this set

- 9618/21 N25 Q3 ▪ 9 marks
- 9618/22 N25 Q1 ▪ 11 marks
- 9618/31 N25 Q1 ▪ 6 marks
- 9618/22 J25 Q1 ▪ 14 marks
- 9618/23 J25 Q1 ▪ 14 marks
- 9618/32 J25 Q1 ▪ 6 marks
- 9618/21 N24 Q1 ▪ 9 marks
- 9618/21 N24 Q4 ▪ 10 marks
- 9618/23 N24 Q1 ▪ 10 marks
- 9618/21 J24 Q1 ▪ 12 marks

Questions in this set

- 9618/22 J24 Q3 ▪ 10 marks

Reading the mark scheme

- **B1** a mark that stands on its own – the point is either made or not.
- **M1** a *method* mark: the right approach, even if the number is wrong.
- **A1** an *answer* mark, and it usually needs its M mark first.
- **C1** a working mark on the way to the answer.
- // separates answers the examiner accepts equally.
- **ecf** = error carried forward: a later part is marked on your own earlier answer.
- **owtte** = "or words to that effect" – the wording need not match.

Question 3

9618/21 N25 ▪ Q3 ▪ 9 marks

A program is needed to manage individual rentals in a car-hire business.
The data items for each rental will be held in a record structure of type `RentalRecord` The programmer has started to define the items that will be needed:

Item	Example value	Comment
RentalID	"AB1234"	a unique alpha-numeric value
CarID	241	a numeric value used as an array index
DisCode	'S'	a letter indicating the type of discount offered
Start	13/06/2025	when the rental starts
Duration	7	the number of days of the rental
Completed	FALSE	TRUE when the car is returned and the rental charge paid

Question 3

(a)(i) Write pseudocode to declare the record structure for type `RentalRecord` **[4]**

(a)(ii) A 1D array `Rental` containing 500 elements is used to store the data for all rental records.

Write pseudocode to declare the `Rental` array. **[2]**

(b) State **three** benefits of using an array of records to store the data for all rentals.

1. 2. 3. **[3]**

Question 3

Item	Example value	Comment
`RentalID`	`"AB1234"`	a unique alpha-numeric value
`CarID`	`241`	a numeric value used as an array index
`DisCode`	`'S'`	a letter indicating the type of discount offered
`Start`	`13/06/2025`	when the rental starts
`Duration`	`7`	the number of days of the rental
`Completed`	`FALSE`	`TRUE` when the car is returned and the rental charge paid

Solution 3

(a)(i) Worked steps

A record type groups fields of different types under one name. Pick the type that matches what each field actually holds.

```
TYPE RentalRecord
    DECLARE RentalID : STRING
    DECLARE CarID : INTEGER
    DECLARE DisCode : CHAR
    DECLARE Start : DATE
    DECLARE Duration : INTEGER
    DECLARE Completed : BOOLEAN
ENDTYPE
```

Solution 3

- `TYPE … ENDTYPE` carries a mark on its own — a bare list of DECLAREs is not a record type.
- A **single** character is `CHAR`, not `STRING`: the discount code is one letter.
- A date has its own type; do not store it as a string.
- A yes/no field is `BOOLEAN`, never an INTEGER flag.

Mark scheme

Pseudocode: TYPE RentalRecord DECLARE RentalID : STRING DECLARE CarID : INTEGER DECLARE DisCode : CHAR DECLARE Start : DATE DECLARE Duration : INTEGER DECLARE Completed : BOOLEAN ENDTYPE

Solution 3

Mark as follows:

1. One mark for `TYPE RentalRecord` and `ENDTYPE` statements
2. One mark for `RentalID` and `CarID` declared correctly
3. One mark for `DisCode` and `Start` declared correctly
4. One mark for `Duration` and `Completed` declared correctly

Solution 3

(a)(ii) Worked steps

```
DECLARE Rental : ARRAY[1:500] OF RentalRecord
```

Solution 3

Both marks are visible in that one line: the **bounds** `[1:500]`, and `OF RentalRecord` — an array of the type you just defined, not of STRING.

Mark scheme

- `DECLARE Rental : ARRAY[1:500] OF RentalRecord`
- One mark per underlined phrase

Solution 3

(b)

Mark scheme

One mark per point:

1. **Different data types** held (for each rental) under a single identifier
2. Allows for iteration through rentals/records // Can access rentals/records in a loop // Can (directly) access a specific rental/record using an (array) index
3. Simplifies the code/program / less code needed / reduces code duplication // Makes code/program easier to understand // Makes testing / debugging easier // **Program** maintenance easier

Question 1

9618/22 N25 ▪ Q1 ▪ 11 marks

Refer to the **insert** for the list of pseudocode functions and operators.

(a) The table contains pseudocode examples.

Each example may contain statements that relate to one or more of:

- selection
- iteration (repetition)
- subroutines (procedures or functions).

Complete the table by placing **one or more** ticks (✓) in each row.

Question 1

Pseudocode example	Selection	Iteration	Subroutine
IF Status = FALSE THEN FOR Count <- 1 TO 20 CALL Reset(Count) NEXT Count ENDIF			
OTHERWISE : Status <- TRUE			
WHILE AllDone() = TRUE			
30 : NextChar <- 'X'			

[4]

Question 1

Pseudocode example	Selection	Iteration	Subroutine

`IF Status = FALSE THEN`
`  FOR Count ← 1 TO 20`
`  CALL Reset(Count)`
`  NEXT Count`
`ENDIF`			

`OTHERWISE : Status ← TRUE`			
`WHILE AllDone() = TRUE`			
`30 : NextChar ← 'X'`			

[4]

Question 1

Expression	Evaluates to
`INT(Result) + 1 > 6`	
`NUM_TO_STR(LENGTH(MonthLetter))`	
`NUM_TO_STR(Result + "3.2")`	
`MID(MonthLetter, MONTH(Birthday) − 2, 1)`	

Question 1

Variable	Example data value	Data type
`Result`	`5.42`	
`MonthLetter`	`"JFMAMJJASOND"`	
`Birthday`	`15/11/2009`	

Question 1

9618/22 N25 ▪ Q1 ▪ 11 marks

Refer to the **insert** for the list of pseudocode functions and operators.

(b) Complete the table by giving the appropriate data type.

Variable	**Example data value**	**Data type**
`Result`	5.42	
`MonthLetter`	`"JFMAMJJASOND"`	
`Birthday`	15/11/2009	

Question 1

[3]

Question 1

9618/22 N25 ▪ Q1 ▪ 11 marks

Refer to the **insert** for the list of pseudocode functions and operators.

(c) Evaluate each expression in the table by using the data values shown in (b).
Write 'ERROR' if the expression contains an error.

Expression	**Evaluates to**
`INT(Result) + 1 > 6`	
`NUM_TO_STR(LENGTH(MonthLetter))`	
`NUM_TO_STR(Result + "3.2")`	
`MID(MonthLetter, MONTH(Birthday) - 2, 1)`	

Question 1

[4]

Solution 1

(a) Worked steps

Judge each extract by what its statements **do**, and remember a single extract can be more than one thing.

Extract	Selection	Iteration	Subroutine
`IF Status = FALSE THEN … FOR Count <- 1 TO 20 … CALL Reset(Count) … NEXT Count … ENDIF`	✓	✓	✓
`OTHERWISE : Status <- TRUE`	✓		
`WHILE AllDone() = TRUE`		✓	✓
`30 : NextChar <- 'X'`	✓		

Solution 1

The rows that carry the marks are the ones doing more than one thing:

- The first is all three at once: an IF, a FOR, and a CALL.
- `WHILE AllDone() = TRUE` loops **and** calls a function in its condition — a function call is a subroutine call wherever it appears.
- Both `OTHERWISE` and `30 :` are branches of a CASE statement, so both are selection.

Solution 1

Mark scheme

- Pseudocode example
- Selection
- Iteration
- Subroutine
- IF Status = FALSE THEN
- FOR Count <- 1 TO 20
- CALL Reset(Count)
- NEXT Count
- ENDIF
- Π
- Π
- Π

Solution 1

- OTHERWISE : Status <- TRUE
- Π
- WHILE AllDone() = TRUE
- Π
- Π
- 30 : NextChar <- 'X'
- Π
- One mark per row

Solution 1

(b) Worked steps

Pick the type from the **example value**, not from the variable's name.

Variable	Example value	Data type
`Result`	`5.42`	`REAL`
`MonthLetter`	`"JFMAMJJASOND"`	`STRING`
`Birthday`	`15/11/2009`	`DATE`

Solution 1

- `5.42` has a fractional part, so REAL, not INTEGER.
- `MonthLetter` sounds like a single character, but the value shown is twelve of them — STRING. This is the row the question is testing.
- A date has its own type in this syllabus.

Solution 1

Mark scheme

- Variable
- Example data value
- Data type
- Result
- 5.42
- REAL
- MonthLetter
- "JFMAMJJASOND"
- STRING
- Birthday
- 15/11/2009
- DATE

Solution 1

- One mark per row

Solution 1

(c) Mark scheme

- Expression
- Evaluates to
- INT(Result) + 1 > 6
- FALSE
- NUM_TO_STR(LENGTH(MonthLetter))
- "12"
- NUM_TO_STR(Result + "3.2")
- ERROR

Solution 1

- MID(MonthLetter, MONTH(Birthday) — 2, 1)
- "S"
- One mark per row
- 4
- October/November
- 2025

Question 1

9618/31 N25 ▪ Q1 ▪ 6 marks

The composite record data type, `ClubMember`, is defined in pseudocode as:
TYPE ClubMember DECLARE Code : INTEGER DECLARE LastName : STRING DECLARE FirstName : STRING DECLARE Telephone : STRING DECLARE JoinDate : DATE DECLARE Fees : REAL DECLARE FeesPaid : BOOLEAN ENDTYPE

(a)(i) Write the pseudocode statement to set up a variable for one record of the composite data type, `ClubMember`. **[1]**

(a)(ii) Write the pseudocode statements to assign the following values to the variable set up in part (a)(i):

- 984632 to `Code`
- `TRUE` to `FeesPaid`

Question 1

[2]

(b)(i) Write the pseudocode statement for the type declaration of `Activity` to hold the names of the available activities:
Badminton, Football, Golf, Snooker, Swimming, Tennis. **[2]**

(b)(ii) Write the new pseudocode statement required to update the declaration of `Choice` in the definition of `ClubMember`. **[1]**

Solution 1

(a)(i)

Worked steps

```
DECLARE Member1 : ClubMember
```

A composite type is used exactly like a built-in one: `DECLARE <name> : <type>`.

Mark scheme

- DECLARE Member1 : ClubMember

Solution 1

(a)(ii) Worked steps

Reach into a record with **dot notation**, one statement per field.

```
Member1.Code <- 984632
Member1.FeesPaid <- TRUE
```

- 984632 is a number, so no quotes.
- `TRUE` is a Boolean literal, so no quotes either — `"TRUE"` would be a string and loses the mark.

Solution 1

Mark scheme

- One mark for each correct answer
- Example answer
- Member1.Code <- 984632
- Member1.FeesPaid <- TRUE

Solution 1

(b)(i) Worked steps

An enumerated type lists its allowed values in brackets, and they are identifiers rather than strings.

```
TYPE Activity = (Badminton, Football, Golf, Snooker, Swimming, Tennis)
```

Solution 1

Mark scheme

- One mark per mark point (Max 2)
- MP1
- TYPE Activity=
- MP2
- (Badminton, Football, Golf, Snooker, Swimming, Tennis)
- Example answer
- TYPE Activity = (Badminton, Football, Golf, Snooker, Swimming,
- Tennis)

Solution 1

(b)(ii) Worked steps

```
DECLARE Choice : Activity
```

Solution 1

The field's type becomes the new enumerated type, so `Choice` can now only hold one of those six values — which is the whole reason for declaring it.

Mark scheme

- DECLARE Choice : Activity

Question 1

9618/22 J25 ▪ Q1 ▪ 14 marks

Refer to the insert for the list of pseudocode functions and operators.

A program is being developed to control the production line in a factory.

(a)(i) Explain the need for different program development life cycles. **[1]**

(a)(ii) Coding is a stage in a program development life cycle.

State **one** consideration that would influence the choice of programming language. **[1]**

Question 1

Expression	Data type
`RIGHT(MachineCode, 4)`	
`Speed * 2.5`	
`NOT Status`	
`IS_NUM(Check)`	

Question 1

Variable	Lower bound	Upper bound
x	0	99
y	0	9

Question 1

9618/22 J25 ▪ Q1 ▪ 14 marks

Refer to the insert for the list of pseudocode functions and operators.

A program is being developed to control the production line in a factory.

(b)(i) Give **three** reasons why adaptive maintenance may be required.

1. 2. 3. **[3]**

(b)(ii) As well as adaptive maintenance, other types of program maintenance may be needed.

Identify **one** other type of program maintenance. **[1]**

Question 1

9618/22 J25 ▪ Q1 ▪ 14 marks

Refer to the insert for the list of pseudocode functions and operators.
A program is being developed to control the production line in a factory.
(c) Pseudocode has been used to design modules for the program to control the production line.
The table shows **four** valid pseudocode expressions.
Complete the table by giving the data type of the evaluated expression.

Expression	Data type
`RIGHT(MachineCode, 4)`	
`Speed * 2.5`	
`NOT Status`	
`IS_NUM(Check)`	

Question 1

[4]

Question 1

9618/22 J25 ▪ Q1 ▪ 14 marks

Refer to the insert for the list of pseudocode functions and operators.

A program is being developed to control the production line in a factory.

(d)(i) State the number of dimensions of `Product` **[1]**

(d)(ii) State the total number of elements in array `Product` **[1]**

(d)(iii) Give the pseudocode declaration for the array `Product` **[2]**

Solution 1

(a)(i) Mark scheme

- One mark for
- Choice of program development cycle depends on the approach/method required to produce the program
- Or
- One mark for a factor that would influence the choice of program development cycle, e.g.
- ▪ Rapid development of program/prototypes required
- ▪ Need for prototypes (at an early stage)

Solution 1

- ▪ Complexity of problem
- ▪ Skills / experience of development team / programmer(s)
- ▪ Budget / Time / Resources available
- ▪ Size of development team
- ▪ Developer / programmer can return to earlier stages / any stage
- ▪ Avoidance of repeating previous stages in development of program
- ▪ How much involvement clients will have
- ▪ Allows the (program) requirements to be changed during program development
- ▪ (Program) requirements agreed at start of development of program
- Max 1

Solution 1

(a)(ii)

Mark scheme

- Which programming language would best be suited to control the production line // Which programming language would best suit the problem being solved

Solution 1

(b)(i)

Mark scheme

- Examples include:
- ▪ Change to (production line) requirements
- ▪ New technology / hardware available (to control production line)
- ▪ Changes made to library modules used
- ▪ Change in relevant legislation
- Max 3

Solution 1

(b)(ii)

Mark scheme

- Perfective // Corrective

Solution 1

(c) Worked steps

An expression's type is decided by the **operator or function**, not by the types going in.

Expression	Data type
`RIGHT(MachineCode, 4)`	`STRING`
`Speed * 2.5`	`REAL`
`NOT Status`	`BOOLEAN`
`IS_NUM(Check)`	`BOOLEAN`

Solution 1

- `RIGHT` returns characters, so a STRING even when they are all digits.
- Multiplying by `2.5` makes the result REAL whatever `Speed` is — this is the row most often answered INTEGER.
- `NOT` and any `IS_…` test yield TRUE or FALSE, so BOOLEAN.

Solution 1

Mark scheme

- 1 mark for each correct row
- Expression
- Data type
- RIGHT(MachineCode, 4)
- STRING
- Speed * 2.5
- REAL
- NOT Status
- BOOLEAN
- IS_NUM(Check)

Solution 1

- BOOLEAN

Solution 1

(d)(i)

Mark scheme

- Two / 2

(d)(ii)

Mark scheme

- 1000

Solution 1

(d)(iii) Worked steps

```
DECLARE Product : ARRAY[0:99, 0:9] OF INTEGER
```

Solution 1

Both marks are in that line: the **two pairs of bounds** for a 2D array, separated by a comma, and the rest of the declaration — the name, the keyword ARRAY and the element type. A 100×10 array indexed from 0 has upper bounds 99 and 9, not 100 and 10.

Mark scheme

- DECLARE Product : ARRAY [0:99, 0:9] OF INTEGER
- 1 mark for correct upper and lower bound
- ▪ [0:99, 0:9
- 1 mark for all other parts of declaration
- DECLARE Product : ARRAY [0:99, 0:9] OF INTEGER

Question 1

9618/23 J25 ▪ Q1 ▪ 14 marks

A program is being developed for the management of a sports centre.

(a)(i) State **two** reasons why the programmer decides to use modules.

1. 2. **[2]**

(a)(ii) State **one** difference between local and global variables. **[1]**

(a)(iii) State **two** benefits of using local variables.

1. 2. **[2]**

Question 1

Expression	Evaluates to
.....(68)	'D'
.....(04/02/2025)	2
.....TRUE	FALSE
.....(..... ("Court1.4Upper" ,.....,.....))	1.4

[5]

Question 1

Variable	Use of variable	Data type
`MemberCount`	stores how many members have used the sports centre each day	
`TotalTakings`	stores the total amount of money taken each day	
`BookingConfirmed`	stores the state of a booking for an exercise class; a booking is either confirmed or **not** confirmed	
`MemberDOB`	to calculate when to send an email with a birthday message to a member	

[4]

Question 1

9618/23 J25 ▪ Q1 ▪ 14 marks

A program is being developed for the management of a sports centre.

(b) The pseudocode design contains a number of expressions.

Complete each pseudocode expression so that it evaluates to the value shown.

Refer to the insert for the list of pseudocode functions and operators.

Expression	Evaluates to
_____(68)	'D'
_____(04/02/2025)	2
_____TRUE	FALSE
_____(_____("Court1.4Upper", ____, ____))	1.4

Question 1

[5]

Question 1

9618/23 J25 ▪ Q1 ▪ 14 marks

A program is being developed for the management of a sports centre.

(c) The table lists some of the variables used in the program.

Complete the table by writing the most appropriate data type for each variable.

Variable	**Use of variable**	**Data type**
`MemberCount`	stores how many members have used the sports centre each day	
`TotalTakings`	stores the total amount of money taken each day	
`BookingConfirmed`	stores the state of a booking for an exercise class; a booking is either confirmed or **not** confirmed	
`MemberDOB`	to calculate when to send an email with a birthday message to a member	

Question 1

[4]

Solution 1

(a)(i) Mark scheme

- MP1
- When a task/module is repeated / reused / called // performed in several places
- MP2
- A specific task can be identified /coded as a module / subroutine / procedure / function.
- MP3
- Reduces complexity of program / code // Program / code is simplified
- MP4

Solution 1

- Module / subroutine / procedure / function already available.
- MP5
- Testing / debugging / maintenance is easier
- Max 2

Solution 1

(a)(ii) Mark scheme

- MP1
- Local variables are used within a module // are accessible only within the module (in which they are declared)
- MP2
- Global variables are accessible to all parts of the program /
- MP3
- Local variables use memory only while the module in is executing //
- Global variables use memory for the whole time the program is running

Solution 1

- Max 1

Solution 1

(a)(iii) Mark scheme

- MP1
- The same variable name can be reused in other parts of the program
- MP2
- Using local variables makes modules self-contained // cannot accidentally change the same identifier outside of the module
- MP3
- Using local variables aids modularisation.
- MP4

Solution 1

- Memory allocated to local variables can be reused when module not in use
- Max 2

Solution 1

(b) Worked steps

Work backwards from the required value: pick the function whose job is exactly that conversion.

- `CHR(68)` evaluates to `'D'` — `CHR` turns a character **code** into the character, and 68 is 'D' in ASCII.
- `MONTH(04/02/2025)` evaluates to 2 — the date functions each pull one part out of a DATE.
- `NOT TRUE` evaluates to `FALSE`.
- `STR_TO_NUM(MID("Court17", 6, 2))` evaluates to 17 — `MID` takes the two characters starting at position 6, giving the string `"17"`, and `STR_TO_NUM` makes it a number.

Solution 1

The last one is the pattern worth remembering: extract with a string function, then convert. Leaving out the conversion returns "17", a string, which is not what was asked for. **Mark scheme**

- Mark as follows:
- Expression
- Evaluates to
- Mark
- CHR(68)
- 'D'

Solution 1

- 1
- MONTH(04/02/2025)
- // -2 + DAY(04/02/2025)
- // -2023 + YEAR(04/02/2025)
- 2
- 1
- NOT TRUE
- // FALSE = TRUE
- FALSE
- 1

Solution 1

- STR_TO_NUM(MID("Court1.4Upper",6,3))
- 1.4
- 2
- Mark Row 4 as follows:
- STR_TO_NUM
- 1 mark
- MID("Court1.4Upper",6,3)
- 1 mark

Solution 1

(c) Worked steps

Choose the narrowest type that can hold every value the variable needs.

Variable	Data type
`MemberCount`	`INTEGER`
`TotalTakings`	`REAL`
`BookingConfirmed`	`BOOLEAN`
`MemberDOB`	`DATE`

Solution 1

- A count of members is always whole, so INTEGER; money has parts of a unit, so REAL.
- A yes/no fact is BOOLEAN, not a STRING holding "yes".
- A date of birth is a DATE — storing it as a string makes it impossible to compare or to take a year from.

Solution 1

Mark scheme

- Mark as follows:
- 1 mark per row
- Variable
- Data type
- MemberCount
- INTEGER
- TotalTakings
- REAL
- BookingConfirmed BOOLEAN
- MemberDOB

Solution 1

- DATE

Question 1

9618/32 J25 ▪ Q1 ▪ 6 marks

Data types can be defined using pseudocode.

The composite record data type, `Departure`, is used to represent flights from Cambridge Airport and is defined in pseudocode as:

TYPE Departure DECLARE FlightNumber : STRING DECLARE Destination : STRING DECLARE FlightDate : DATE DECLARE Gate : STRING DECLARE Airline : STRING ENDTYPE

A variable, `Flight1`, is declared in pseudocode as:

DECLARE Flight1 : Departure

(a) Write **pseudocode** to store the following details to `Flight1`:

Question 1

Field	Data
`FlightNumber`	SB2789
`Destination`	Dublin
`FlightDate`	30/07/2025
`Gate`	N03
`Airline`	Cambridge Airways

Question 1

[3]

Question 1

Field	Data
FlightNumber	SB2789
Destination	Dublin
FlightDate	30/07/2025
Gate	N03
Airline	Cambridge Airways

Question 1

9618/32 J25 ▪ Q1 ▪ 6 marks

Data types can be defined using pseudocode.
The composite record data type, `Departure`, is used to represent flights from Cambridge Airport and is defined in pseudocode as:

```
TYPE Departure
    DECLARE FlightNumber : STRING
    DECLARE Destination : STRING
    DECLARE FlightDate : DATE
    DECLARE Gate : STRING
    DECLARE Airline : STRING
ENDTYPE
```

A variable, `Flight1`, is declared in pseudocode as:

```
DECLARE Flight1 : Departure
```

(b)(i) Write a **pseudocode** statement to declare `GateID` to hold the identity codes for the airport gates:
N01, N02, N03, W01, W02, W03, W04 **[2]**

Question 1

(b)(ii) Write the new **pseudocode** statement required to replace the declaration of `Gate` in `Departure`. **[1]**

Solution 1

(a) Worked steps

A record's fields are reached with a **dot**, and each is assigned separately.

```
Flight1.FlightNumber <- "SB2789"
Flight1.Destination  <- "Dublin"
Flight1.FlightDate   <- 30/07/2025
Flight1.Gate         <- "N03"
```

Solution 1

Three marks: using the `Flight1` **variable with the field names given**; the **string values** assigned correctly; and the **date value** assigned correctly.
The date is the mark that separates the answers. `FlightDate` is of type DATE, so its value is **not in quotation marks** — `"30/07/2025"` is a string and a type error. A DATE literal is written bare, and that is what lets `DAY()`, `MONTH()` and `YEAR()` work on it later.

Solution 1

Everything else is a STRING, including `Gate` and `FlightNumber`: they are identifiers, never arithmetic, and they contain letters. **Mark scheme**

- One mark for each mark point
- MP1
- Use of correct Flight1 variable with given field names
- MP2
- Correct assignments of four string data values
- MP3
- Correct assignment of date data value
- Example answer
- Flight1.FlightNumber <- "SB2789"
- Flight1.Destination <- "Dublin"
- Flight1.FlightDate <- 30/07/2025
- Flight1.Gate <- "N03"
- Flight1.Airline <- "Cambridge Airways"

Solution 1

(b)(i) Worked steps

An **enumerated** type lists the values a variable may take, as identifiers rather than strings.

```
TYPE GateID = (N01, N02, N03, W01, W02, W03, W04)
```

Solution 1

Two marks: the `TYPE GateID =` heading, and the bracketed list with **every** gate present and correctly spelled.
Three points of form:

- **No quotation marks.** `"NO1"` is a string literal; `NO1` is an enumerated value, and they are different types.
- **Round brackets**, values separated by commas.
- **No `ENDTYPE`.** An enumerated type is a one-line declaration, unlike a record.

Solution 1

The reason for defining it at all is what part (b)(ii) then uses: once `GateID` exists, a field declared `: GateID` can only hold one of those seven values, which no STRING field could guarantee. **Mark scheme**

- One mark per mark point
- MP1
- TYPE GateID =
- MP2
- (N01, N02, N03, W01, W02, W03, W04)
- Example answer
- TYPE GateID = (N01, N02, N03, W01, W02, W03, W04)

Solution 1

(b)(ii) Worked steps

Once an enumerated type exists, a field that can only hold one of its values should be declared **of that type** rather than as a string or an integer.

```
DECLARE Gate : GateID
```

Solution 1

One mark, and the whole question is in the type name on the right. The point of defining `GateID` in part (b)(i) is that the compiler can then reject anything that is not a real gate: a `STRING` field would happily accept `"platform 9"` or a typo, and an `INTEGER` field would accept 4000.

Note what does **not** change — the field is still called `Gate`, and it is still declared with `DECLARE` inside the same record. Only the type after the colon is different.

Mark scheme

- DECLARE Gate : GateID

Question 1

9618/21 N24 ▪ Q1 ▪ 9 marks

Refer to the **insert** for the list of pseudocode functions and operators.
A program will calculate the tax payable based on the cost of an item.
Calculations will occur at many places in the program and these involve the use of one of three tax rates.
Tax rate values represent a percentage. For example, a tax rate value of 5.23 represents 5.23
Tax rate values are used at several places within the program. One example is given in pseudocode as follows:
HighRate <- FALSE CASE OF ItemCost <= 50 : TaxRate <- 3.75 // tax rate of 3.75<= 200 : TaxRate <- 5.23 // tax rate of 5.23> 200 : TaxRate <- 6.25 // tax rate of 6.25HighRate <- TRUE ENDCASE TaxPayable <- ItemCost * TaxRate // tax payable

Question 1

(a) The pseudocode contains a logical error.
Identify the error **and** suggest a correction. **[2]**

Question 1

Variable name	Data type
`HighRate`	
`TaxPayable`	

Question 1

9618/21 N24 ▪ Q1 ▪ 9 marks

Refer to the **insert** for the list of pseudocode functions and operators.
A program will calculate the tax payable based on the cost of an item.
Calculations will occur at many places in the program and these involve the use of one of three tax rates.
Tax rate values represent a percentage. For example, a tax rate value of 5.23 represents 5.23
Tax rate values are used at several places within the program. One example is given in pseudocode as follows:

```
HighRate <- FALSE CASE OF ItemCost <= 50 : TaxRate <- 3.75 // tax rate of 3.75<= 200 : TaxRate <- 5.23 // tax rate of 5.23> 200 : TaxRate <- 6.25 // tax rate of 6.25HighRate <- TRUE ENDCASE TaxPayable <- ItemCost * TaxRate // tax payable
```

(b)(i) Identify a more appropriate way of representing the tax rate values in the final program. **[1]**

Question 1

(b)(ii) Describe the benefits of your answer to part (b)(i) with reference to this program. **[3]**